Complete computational sequence characterization of mobile element variations in the human genome using meta-personal genome data

Yaroslava Girilishena

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

Abstract

While a large number of methods have been developed to detect such types of genome sequence variations as single nucleotide polymorphisms (SNPs) and small indels, comparatively fewer methods have been developed for finding structural variants (SVs) and in particular mobile elements insertions (MEIs). Moreover, almost all these methods can detect only the breakpoints of an occurred SV, sometimes with approximation, and do not provide complete sequences representing the SVs.

The main objective of our research is to develop a set of computer algorithms to provide complete genome sequence characterization for insertional structural variants in the human genomes via local *de novo* sequence assembly or progressive assembly using discordant and concordant read pairs and split-reads. An essential component of our approach involves utilizing all personal genome data available in the public domain vs. the standard way of using one set of personal genome sequences.

The developed tool is the first system that provides full sequence characterization of SVs. Overall, the characterization success rate for Alu is 75.03% with the mean of discordant and split-reads higher than 94 reads. For SVA, it is 71.43% with the threshold of 363 reads. And for L1 the values are 77.78% and 355 respectively.

The results showed that the SV characterization depends on the allele frequency and is influenced by the repetitiveness of flanking regions. Therefore, addressing these problems is a key to further improvements.

Acknowledgements

I would like to express my sincere gratitude to Professor Sheridan Houghten, my research supervisor, for her patient guidance, enthusiastic encouragement and useful recommendations for this research. She has made this stressful period more enjoyable and curious. My very great appreciation is offered to Professor Ping Liang for his valuable and constructive suggestions during the development of this research, for attention to details, constant availability, and great dedication to the work. It was a wonderful experience to conduct the research under their supervision.

Special thanks are given to students from the Biology department Zakia Dahi for providing the input data for testing and Jina Nanayakkara for helping in understanding the biological background of the problem.

I would like to thank the Compute Canada/SHARCNET organization for providing the hardware resources to store the enormous amount of data and run the time-consuming program and for their technical support in troubleshooting.

Finally, I wish to thank all my family, especially my parents, for their constant support, motivational speeches, and encouragement throughout my study. Even though they are far away, I could always feel how they pushed me forward to excel and succeed. I would also like to acknowledge the support provided by my husband, Illya Bakurov, for an incredible driving force that made me keep going in the right direction, and for cheering up when bugs happened.

I hope that this work will encourage other students to continue research in this area and contribute to the Bioinformatics community.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Personal genomics

The first steps towards personal genomics were made by the International Human Genome Project (1990-2003) and Celera teams, who completed the human genome sequencing and determined the sequence of chemical base pairs which make up human DNA [38, 74]. They made the database publicly available for further investigation and improvements.

The Genome Reference Consortium Human genome build 38 (GRCh38) is the most recent build of the reference genome. It was constructed from reference sequences of different individuals, while the previous builds were from one individual's genome. The reference genome has been improved over time. However, it still represents one copy of the genome, thus, it is not able to reflect the high level of genome diversity.

With the development of new technologies, such as Next-Generation DNA sequencing (NGS), large-scale genome sequencing became cost effective for individuals, and we can revolutionize our approach to individual health care using personal genomics as a basis of precision medicine [80, 50].

The 1000 Genomes Project has been set up to re-sequence at least a thousand individuals' genomes from around the world with the aim of providing a comprehensive map of human genetic variation for more accurate disease studies [2]. In the final third phase, the researchers reconstructed the genomes of 2,504 individuals from 26 populations in Africa, East Asia, Europe, South Asia, and the Americas. Over 88 million variants including 84.7 million single nucleotide polymorphisms (SNPs), 3.6 million short insertions/deletions (indels), and 60,000 structural variants (SVs) were characterized [2].

For variant discovery, the 1000 Genomes Project integrated 24 sequence analysis tools and classifiers. The detailed information about the implementation and results can be found in supplementary materials [1].

The data from the 1000 Genomes Project has been used to correct errors and fill gaps in the new version of the reference genome. However, the reference genome still has a limited representation of genomic diversity, thus, making the discovery and characterization of structural variants very challenging.

The goal that is still being pursued is to create tools and databases for understanding the genetic factors in human diseases that can improve earlier-stage diagnosis, drug prescription, and risk assessment, as well as provide information about human evolution and how individuals' differences are reflected in their genomes.

1.2 Importance of genomic variants

When the extensive presence of structural variants was recognized in the human genome, such areas of biology as association studies, cancer genomics, and molecular evolution started to discover the impact of structural variants in each particular field. It was determined that diseases such as autism and Parkinson's disease are caused by changes in gene dosage (copy number variations). Besides, the analysis of structural variants has led to a better understanding of how the genome has been shaped throughout evolution [55, 31].

Since individuals' disease susceptibility and treatment response are reflected in genomic variants, the key to the personal genome data usage is to efficiently and accurately detect them.

1.3 Structural variants. Problems

The variations between human genomes consist of single nucleotide polymorphisms (SNPs) which are variations at a single nucleotide level, small insertions or deletions (indels) that are less than 50 base pairs (bp) long, and structural variants (SVs) which are large genome changes that are longer than 50 bp. A base pair (bp) is a unit of two nucleotide bases bound to each other. They are the building blocks of the DNA double helix.

Personal genome analysis includes the sequencing of an entire genome and accurate detection of genomic variants that are present in the given genome. Variant discovery depends on the completeness and diversity of the reference genome. Thus,

current data analysis strategies produce many false-positive and false-negative results in variant detection, especially structural variants.

The genomic variants are inherently different from the reference genome due to sequence diversity - some sequences that are present in an individual's genome may be missing in the reference since the individuals from whom the reference genome was constructed did not have it. Therefore, the current reference genome does not reflect human genome diversity and variants' evolutionary state.

The single nucleotide polymorphisms (SNPs) and small insertions and deletions (indels) can be relatively easily identified by examining the mismatches and small gaps in their alignments with the reference. However, the detection of SVs, which involve sequence changes or rearrangements at a much larger scale, is much harder due to the short length of sequencing reads provided by most next-generation sequencing (NGS) platforms.

There are many types of structural variants that differ in pattern and length. The most common ones are copy number variations (large insertions and deletions), inversions, translocations, and mobile elements insertions.

Currently, all methods for SV detection rely on identifying sequence reads that show a discordant pair-end and/or a split mapping pattern, i.e. sequence features not present in the reference genome. The identified SVs often lack a base-pair resolution and mostly lack a full sequence characterization. Only the breakpoints of an occurred SV are detected, sometimes with an approximation. However, the complete sequence characterization for SVs is important for prediction of the functional impact of the variant.

Last, but not least, current strategies suffer from the fact that each test genome is analyzed individually without utilizing accumulated genomic sequence and variation data at the human population level. Most recent methods are described in details in the Background section.

These are the major problems we are facing today when dealing with SV discovery.

1.4 Main objectives

The current algorithms for detecting different types of structural variants (SVs) take advantage of various signals provided by NGS mapping algorithms. However, the identification of mobile element insertions (MEIs) with NGS data is not successful because mobile elements are highly repetitive DNA sequences that are difficult to align against the reference genome with commonly used mapping strategies.

Therefore, the main objectives are to develop a set of computer algorithms to provide complete genome sequence characterization for insertional structural variants in human genomes using discordant, concordant and split-reads of the utilized personal genome data. In this work, we focus on MEIs as they represent the most common SV types by number and they are the most difficult to characterize.

Each SV sequence will cover not only the full sequence associated with the specific variant but also associated local rearrangements. For example, in the case of an MEI, the complete SV sequence includes the inserted mobile element and also the target site duplications (TSDs) - a short sequence that is duplicated and attached to both sides of an insertion. Two sequences representing 600 bp (base pairs) of flanking regions on both sides will also be included.

The success of characterization depends on how long the insertion sequence is and how much quality data is available. Therefore, an essential component of our approach involves utilizing all personal genome data available in the public domain.

The algorithm presented in this research is capable of providing full sequences or at least critical breakpoint sequences for the majority of non-reference SV alleles.

1.5 Organization of thesis

The second chapter introduces the main Bioinformatics definitions and concepts that are necessary for understanding and processing the data for this work. In this chapter, the latest works in the area are discussed and the most effective tools are described as well. The data set, the approach, the algorithm, and validation are described in detail in the third chapter. The achieved results and problems we encountered are presented in chapter four. In the concluding chapter, we summarize the work that has been done, discuss advantages and drawbacks of the approach and further research steps are outlined.

Chapter 2

Background

This chapter covers the Bioinformatics terms and concepts that describe the proposed problem from the biological side and provides a deeper understanding of the nature of processes. We discuss DNA structure, sequencing strategies, the assembly problem and approaches to solve it. In detail, we describe structural variants, the signatures they leave in genomes and by which they can be detected, cover different types of the most challenging mobile element insertions, and provide an overview of current tools for SV discovery. More computational approaches to biological problems can be found in [63], [17], and [61].

2.1 DNA molecule

In 1953 James Watson and Francis Crick determined that the structure of DNA is a double-helix polymer, a spiral consisting of two DNA strands wound around each other (Figure 2.1). Single strands of DNA are (A, C, G, T)-quaternary sequences, with the four letters denoting the respective nucleic acids (bases): adenine (A), guanine (G), cytosine (C), and thymine (T). Strands of DNA are oriented; thus, AACG is distinct from GCAA. One end of the sequence is denoted as 3' and the other as 5'. Furthermore, in nature DNA is ordinarily double stranded: each sequence, or strand, occurs with its reverse complement. Only strands of opposite orientation can form a stable duplex. The Watson-Crick complement of a DNA sequence is another DNA sequence which replaces all occurrences of A with T, and vice versa, replaces all occurrences of C with G, and vice versa, and also switches the 5' and 3' ends. To obtain the reverse complement of a strand of DNA: (i) reverse the order of the letters and (ii) substitute each letter with its complement. For example, the reverse complement of AACGTG is CACGTT. The double strand resulting from adjoining

these reverse complementary strands in opposite orientations is:

AACGTG
TTGCAC

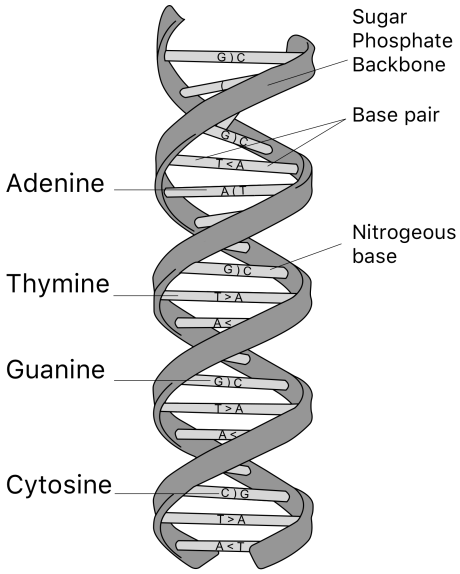

Figure 2.1: **DNA structure.** Two DNA stands that consist of four nucleotide bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Each base with its reverse complement creates a base pair.

2.2 Genome sequencing

Genome sequencing is a determination of the physical order of DNA nucleotides, or bases, in a genome. The haploid human genome is made up of over 3 billion of these genetic letters, therefore, genome sequencing is a complex task. A single sequencing reaction can handle 300 to 1000 nucleotides of a sequence. Therefore, a genome must be sequenced in fragments and assembled after. DNA sequencing is used to determine the sequence of individual genes, full chromosomes or entire genomes.

The foundation for sequencing DNA was laid by the work of Fred Sanger who by 1955 had completed the sequence of all the amino acids in insulin.

In 1977 Fred Sanger with colleagues developed the chain termination method or Sanger sequencing method. This method was used by the Human Genome Project to determine the sequences of small fragments of human DNA. These fragments were each around 900 bp long. To assemble the larger regions (and eventually, the entire chromosome), fragments were assembled based on their pair-wise overlaps.

The first steps towards understanding the human genome were made by the International Human Genome Project (IHGP) under the direction of Francis Collins in 1990. Later, the Celera Genomics corporation led by Dr. Craig Venter entered the field and was able to conduct their research even faster. As a result, by 2001 the blueprint of the human genome was created [17].

The IHGP used a hierarchical sequencing technique which relies on direct sequencing (or primer walking), while the Celera team developed and applied a faster whole-genome shotgun sequencing technique. Although direct sequencing is very accurate and minimizes assembly problems, it is very slow. A genome is broken into long fragments (up to 10^6 bases each) and cloned into specialized vectors - bacterial artificial chromosomes (BACs). After that, fragments from the BACs are cloned into plasmid vectors. Then these fragments are sequenced by primer walking and assembled into a contig that represents the continuous sequence of the BAC. The assembly part was a straightforward task since the relationships between BAC fragments were known.

The Celera team developed a new technique without using BAC. In this technique, the genome is broken up into small fragments (a few thousand bases each) from random positions in the genome that are cloned into plasmid vectors. The sequencing is done for each end of each clone using a single primer complementary. The fragments overlap each other, thus it is possible to assemble them back together. The increased speed is achieved by running multiple sequencing processes in parallel. However, since there is no information about the positions of fragments, the assembly algorithms need to be very accurate.

The introduction of next-generation sequencing (NGS) technologies has changed the way we approach genomics research. New sequencing methods were established, such as (i) pyrosequencing (Ronaghi *et al.*, 1996) conducted by 454 sequencing technology; (ii) sequencing by synthesis by Illumina; and (iii) sequencing by ligation by ABI SOLiD. All of them are high-throughput and low-cost technologies in comparison with the previous Sanger sequencing method. However, they produce much shorter reads (around 400-500 bp for pyrosequencing, 200 bp for Illumina, and 35 bp for ABI SOLiD) and the error rate is higher than in the traditional capillary sequencing. Moreover, NGS technologies have made it possible to sequence thousands of human genomes which gives us an opportunity to build better genome assemblies - pan-genomes [43] - and thus conduct a better analysis.

An important stage in genome sequencing is the assembly of reads - to make contiguous sequences, called contigs, from randomly picked fragments from the sample

that will represent DNA. The traditional assembly of shotgun reads relies on the overlap-layout-consensus approach [65] where all the reads are compared to each other in a pair-wise fashion. Unfortunately, this approach is not suitable for very short reads ($\leqslant$ 50bp). Since they are produced in large quantities and at bigger depth coverage, the amount of data to compute becomes enormous. Whereas long reads with long overlaps almost remove the ambiguity of the alignment, short reads within repeats are very difficult to examine.

These drawbacks of integrating the old approach in new technologies have led to the development of *de novo* assembly tools that aim to deal with these very short reads.

2.2.1 Sequencing Methods

2.2.1.1 Basic Methods

The basic sequencing methods involve several stages: (i) extracting DNA, (ii) breaking it into fragments, (iii) cloning the fragments and sequencing their tips, (iv) assembling the obtained DNA sequences into one long consensus.

Maxam-Gilbert sequencing

The Allan Maxam and Walter Gilbert sequencing (also known as chemical sequencing) method is based on chemical modification of DNA. The method requires radioactive labeling at one 5' end of the DNA and allows purified samples of double-stranded DNA to be used without further cloning [51]. However, due to the technical complexity and the use of radioactive labeling, this method was not widely used and was substituted by the Sanger sequencing method.

Sanger sequencing

For many years the Sanger sequencing method was used to produce completed genomes (The International Human Genome Sequencing Consortium, 2001; The Mouse Genome Sequencing Consortium, 2002). However, there are some drawbacks, such as dependency on clone libraries, low throughput, and the very high cost of sequencing - $100 million per one human genome.

Shotgun sequencing

Shotgun sequencing was designed to analyze longer DNA sequences from around 1000 bp, up to entire chromosomes. The traditional method consists of breaking the DNA sample into random fragments, sequencing the fragments, and assembling them back together computationally based on their overlaps [8].

Shotgun sequencing works well on short sequences without repetitive regions, such as bacterial genomes, but in repeat-rich genomes, such as mammalian, the method needs improvements.

Clone-by-clone sequencing

Before sequencing starts, a map of each chromosome of the genome is made.

After that, the genome is broken up into relatively large chunks, called clones, about 150,000 bp long. Using genome mapping techniques, it determines the positions in the genome to which these clones belong. Then, these chunks are inserted into bacterial artificial chromosomes (BACs) and put inside bacterial cells to grow. With every division of bacteria, lots of identical copies of chunks are produced.

Subsequently, each clone is cut into smaller, overlapping pieces of the right size for sequencing (about 500 bp each). Finally, the pieces are sequenced and the overlaps are used to reconstruct the sequence of the whole clone.

Clone-by-clone sequencing was the preferred method during the Human Genome Project, which was completed in 2001.

However, this method is more expensive than other sequencing methods and more time consuming due to the clone generation and genome map building. Another drawback of clone-by-clone sequencing is that it cannot sequence long repetitive sections.

Whole-genome shotgun (WGS) sequencing

The other strategy introduced by Venter *et al.* [74] is the whole-genome shotgun method. During the WGS process, the genome is first broken up into small pieces. Then, the pieces are sequenced and assembled into the full genome sequence. There is no clonal map, and the assembly process is more complex. All reads are processed

together and the algorithm needs to handle the misassembling of non-consecutive regions of the genome.

2.2.1.2 Next Generation Sequencing

The advent of NGS technologies reduced the cost of sequencing the genome from $100 million in 2001 to $1000 in 2011. NGS technologies work in a massively parallel fashion, offering advantages in throughput, scale and speed. They produce much smaller fragments, called short reads, than traditional Sanger sequencing, although with a higher error rate.

There are three platforms that are commonly used now for massively parallel DNA sequencing read production: the Roche 454 FLX, the Illumina Solexa Genome Analyzer, and the Applied Biosystems SOLiD^{TM} System.

Roche/454 FLX Pyrosequencer

The 454 sequencer was the first NGS sequencer brought to the market in 2005. Since then, many improvements have been made and now their read length is ahead of others. As well, the 454 technology does not require cloning of the environmental samples. It uses an alternative sequencing technology known as pyrosequencing that is based on the light detection principle [47].

The 454 pyrosequencing approach can determine more than 300,000 sequences at once for the same price as 96-192 sequencing reactions performed using traditional chemistries [49].

Illumina Genome Analyzer

The Illumina Genome Analyzer (originally Solexa) [11] uses bridge amplification and sequencing by a synthesis method to produce reads. The read length is much shorter than 454 reads, but the cost per base is much lower and the throughput is higher.

In this work, we used the sequence data from the standard Illumina 2000/25000 platform.

Applied Biosystems SOLiD^{TM} Sequencer

The SOLiD platform implements the same method for sample preparation as the 454 sequencer. DNA is fragmented and attached to adapter sequences, denaturated and linked to beads. The emulsion PCR is used to amplify the sequences around the beads, which then attached to a sequence inserted into the reaction cell. Unlike the other platforms, SOLiD integrates sequencing by ligation to sequence amplified fragments, colourspace analysis for base calling, and two-base encoding principle for quality evaluation by aligning reads to a reference genome.

The read length is defined by the user and can be between 25-35 bp, and each sequencing run produces 2-4 Gb of DNA sequence data.

2.2.2 Mate pairs / Paired-ends reads

There are two sequencing strategies that can generate two reads from both sides of a segment of DNA at an approximately known distance: mate pairs and paired-end reads. To generate paired-end reads, first, genomic DNA is fragmented into short segments (less than 300 bp), then each segment is sequenced from both ends (as shown in Figure 2.2).

Figure 2.2: **Paired-end reads technology.** In the beginning, DNA is fragmented and fragments shorter than 800 bp are selected. Then, the ends are repaired, and paired-end adapters are added. The resulting fragments are amplified by Polymerase Chain Reaction (PCR), purified and sequenced from both eds.

To create mate pairs, the inserts of a specific size from fragmented DNA are circularized and linked with the help of an internal adaptor. Then, each circularized fragment is randomly cut, and the segments that contain the adapter are kept and subject to sequencing like in the case of pair-end libraries. At the final stage, the mate pairs are generated by sequencing around the adapter (as shown in Figure 2.3).

The advantages of paired-end reads are that (i) they can detect repetitive structural variants, and (ii) they can precisely define locations of structural variants. However, paired-end reads rely on independent mapping of each read which causes problems in regions containing repeats. When dealing with complex regions, the resolution is limited due to a small span between reads, while a large span limits the resolution between break points.

From a computational perspective, there is no difference between these two types of reads [55]. In this work, we operate with paired-end data.

Paired-end reads or mate pairs have been used for structural variant (SV) discovery [73, 36]. In this approach, paired-end reads from a donor genome are mapped to the reference genome. If reads in a pair are mapped at a substantially different distance than they were generated, or with incorrect orientation, then it can be a signal for a structural variant. The technique that extracts such reads is called paired-end mapping (PEM). It is successfully used to discover SVs at a higher resolution than array-based methods [55].

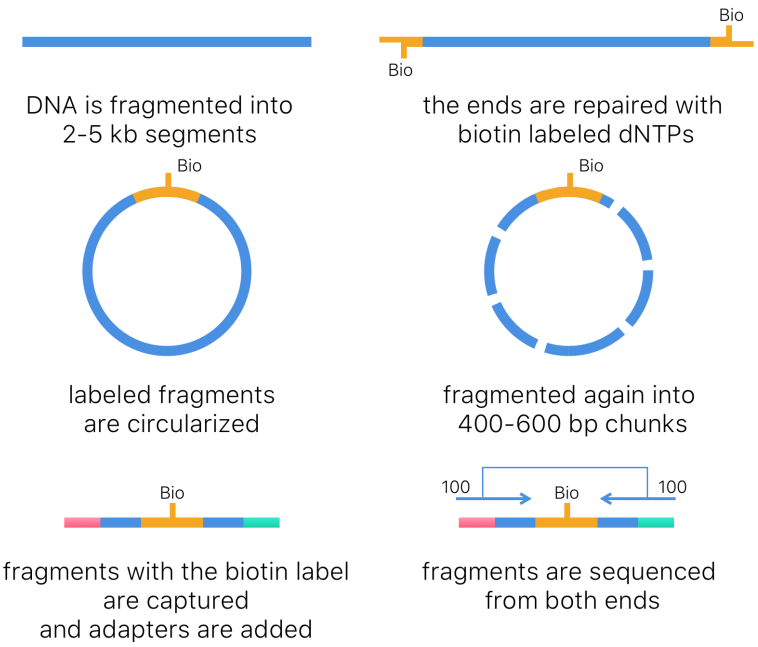

Figure 2.3: **Mate pairs technology.** In the beginning, DNA is fragmented into segments of 2-5 kb length, and the adapter is added to both ends. Then, these fragments are circularized and fragmented again into smaller chunks (400-600 bp). The labeled fragments are captured and ligated with adapters. In the end, these fragments are sequenced from both ends.

2.2.3 Applications and Importance

Sequencing the genome can reveal almost all hidden information and be the guide to understanding it. Information obtained from sequencing the genome allows us to identify genes, phenotypes, a risk of genetic diseases, and drug responses. Moreover, DNA sequencing allows us to study the evolution of different organisms and how they are related to each other.

With the advent of new technologies, DNA sequencing became cost effective and able to be conducted in a shorter period of time, which creates a potential for personal genomics.

For example, The Cancer Genome Atlas project, supported by the National Human Genome Research Institute (NHGRI) and the National Cancer Institute, is using DNA sequencing to unravel the genomic details of 30 cancer types. Another National Institutes of Health program examines how gene activity is controlled in different tissues and the role of gene regulation in disease. Using DNA sequencing, researchers can examine the development of common and complex diseases, such as diabetes,

heart disease, and inherited diseases.

2.3 The Fragment Assembly Problem

Genome assembly is the process of reconstructing the unique single and contiguous sequence of a DNA molecule by using smaller sequences from random locations of the genome.

The DNA fragment assembly problem is one of the last steps in DNA sequencing [37]. It involves finding the overlaps between fragments (or reads), merging the correctly overlapping reads into contigs (single continuous sequences), and assembling the contigs into scaffolds (or supercontigs) that define the contigs' order and orientations as well as the gaps (missing bases) between contigs.

Unfortunately, the initial sequences produced by NGS platforms contain errors. This means that sequences that overlap do not match up perfectly. Moreover, more than 50% of the human genome consists of repeated regions. Therefore, it is difficult to determine whether two sequences truly overlap or whether it is just a random coincidence.

First generation reads were long enough (500 bp to 1000 bp) to avoid this problem. They could cover the large insertions and the error rate was very low. In comparison, the NGS reads are much shorter, e.g., about 400 bp (the 454 platform), 100-200 bp (Illumina and SOLiD), and they provide less information. The coverage should be high enough for successful assembly. Therefore, the target genome is over-sampled with overlapping short reads from random positions.

Coverage (read depth) is the average number of reads representing a given nucleotide in the genome. It can be calculated from the length of the original genome (G), the number of reads (N), and the average read length (L) as $C = (N * L)/G$.

Shotgun sequencing is used along with NGS technologies for DNA sequencing. The shotgun method works by making several copies of the sequence and dividing them into many small pieces called reads. The overlapping reads are merged together creating a single continuous sequence called a contig.

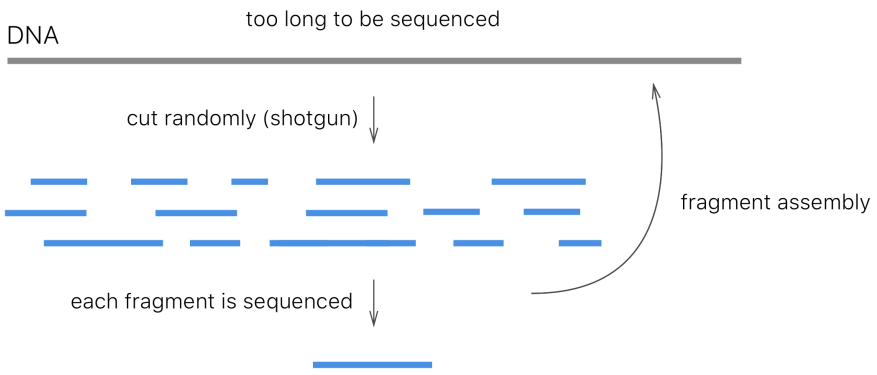

Figure 2.4: **Shotgun sequencing overview.** A DNA sequence is cut randomly into short fragments that can be sequenced. After that, the sequenced fragments, called reads, are assembled to reconstruct the initial DNA sequence.

The success of sequencing technologies rests on the ability to assemble the fragments to produce continuous sequences for an entire chromosome. The variety of sequencing platforms leads to a corresponding variety of the fragments they produce, including the number and lengths of the fragments.

The main criteria for assemblies are the size and accuracy of produced contigs and scaffolds. The size is represented by statistics including maximum and average length, combined total length, and N50 value.

The N50 value is the length of the shortest contig in the set of contigs whose combined length represents at least 50% of the genome. Larger N50 values indicate larger contigs. The N50 value defines the assembly quality. However, N50 values from different assemblies cannot be compared, unless they have the same combined length value. The accuracy of an assembly is difficult to measure since there is no trusted reference that would cover all regions.

Unfortunately, there is no fully functional DNA assembler that would work with any input dataset. Assembly algorithms are challenged for several reasons:

1. Different sequencing platforms produce reads with different characteristics such as the length of the reads, noise distribution, and error rate.

2. The complexity of genomes is defined by the repeating factors. Since there are repeated regions that are larger than read size, it is difficult to resolve them. Moreover, repeats can occur inside other repeats. Repeats longer than the reads can be resolved by spanning paired ends. NGS data with short reads have less capacity to resolve genomic repeats but higher coverage increases the chance of

spanning short repeats. The sequencing errors make it even harder to resolve repeats, thus chimeric assemblies occur frequently in polymorphic repeats.

3. Some assembly methods do not scale to large genome sequencing projects.

4. The implementation of an assembly algorithm is very complex. For large genomes, it requires high-performance computing platforms. The success depends on heuristics that help overcome repeat patterns in real genomes, data errors, and the physical limitations of computers.

The main approaches to the fragment assembly problem are:

1. Comparative (re-sequencing) approach: the sequence of a closely related organism is used to guide the assembly.

2. *De novo* assembly: reconstructing a genome that has never been sequenced before, which is considered to be within a set of NP-hard problems. The main strategies are the greedy graph-based, the overlap-layout-consensus (OLC) [5, 74, 57] and the de Bruijn graph approach [65]. In general, the assembly task is associated with a graph reduction problem which belongs to NP-hard class of problems. Therefore, assemblers rely on heuristics to remove redundancy, repair errors, reduce complexity, and simplify the graph.

These two approaches are not exclusive. Even if a reference genome is used, there are regions (such as large insertions) that are significantly different from the reference, thus *de novo* assembly should be used to reconstruct them.

2.3.1 Greedy graph-based approach

The greedy graph-based approach was implemented in the first assembler used by NGS technologies.

The approach is very simple and consists of one basic operation: to a given read or contig, add another read or contig based on the next highest-scoring overlap (the number of matching bases). This step is repeated until no more operations are possible. This way the contigs are extended greedily. The algorithm can get stuck at local maxima and are not able to produce complete assemblies when dealing with complex situations. However, it is very fast.

The greedy graph-based approach was used by such assemblers as SSAKE [76], SHARCGS [18], and VCAKE [30]. All of them applied filters to avoid false-positive overlaps that could produce chimeric sequences.

2.3.2 The Overlap-Layout-Consensus approach

The overlap-layout-consensus approach was typically integrated by Sanger-data assemblers.

The OLC approach uses an overlap graph where nodes represent reads and edges represent overlaps between reads. The process has three phases:

1. *Overlap:* Compare each pair of reads using all-against-all, pairwise read comparison and find the best overlap between the suffix of one read and the prefix of the other. The overlap must exceed some predefined threshold used to help eliminate non-significant, coincidental overlaps. This overlap between the two fragments is considered the score. The pairs that do not share a common sequence that is long enough are filtered out.

   Normally this process is accomplished with a dynamic programming algorithm applied to semi-global alignments such as Smith-Waterman (Smith and Waterman 1981).

Figure 2.5: **Overlap phase.** Each pair of reads is compared in a pairwise fashion, and the best overlap between them is found.

2. *Layout:* Determine an ordering of the contigs based on the overlap scores. It is ideal to maximize this overlap score. This is the most challenging part of the process (according to Pevzener *et al.*, 2000, it is an NP-hard problem). Then, merge overlapping reads into contigs.

Figure 2.6: **Layout phase.** The order of contigs is determined, and the overlapping sequences are merged.

3. *Consensus:* Determine the most likely DNA sequence based on results from the layout phase and merge overlapping contigs onto a single continuous sequence called a scaffold. This can be done by finding a Hamiltonian path which traverses all nodes in the graph. A sufficient number of reads are required to ensure a statistically significant consensus. In the end, the reading errors are corrected.

Figure 2.7: **Consensus phase.** Overlapping contigs are merged into a scaffold.

The Newbler software distributed by the 454 sequencing platform used an improved version of the OLC approach. Edena and Shorty assemblers applied the OLC approach to the short reads from the Illumina and SOLiD platforms.

However, the overlap-layout-consensus approach is not suitable for short reads since the time required in the overlapping phase is proportional to the square of the number of reads. Thus it is not efficient to run this algorithm with a big dataset of reads.

2.3.3 De Bruijn graph approach

Indury and Waterman (1995) [27] were the first to use a sequence graph to represent an assembly for the sequencing by hybridization technique. Every detected nucleotide sequence of length k - a k-mer - was considered as a node and the edge between two nodes represented an overlap between two k-mers. The contigs were created by chaining the overlapping k-mers.

Later, an improved method was proposed by Pevzner *et al.* [65] who introduced a different formalization of the sequence graph called a *de Bruijn* graph. Currently, de Bruijn graphs are the most commonly used approach for NGS data assembly problem. Such assemblers as EULER-SR [65], Velvet [81], SOAPdenovo [45], SPAdes [10], MEGAHIT [40], and ABySS [68] are designed based on de Bruijn graphs with slight

modifications like parallelization and sparse method (where k-mers are combined into groups instead of being stored independently).

A de Bruijn graph is defined by Pevzner [63] as follows. Suppose that $S = \{s_1, s_2, ..., s_n\}$ is a set of reads and S_{l-1} is a set of vertices that contains all $(k-1)$-mers from the set of reads. Two $(k-1)$-mers i and j are joined by a directed edge, if S_l contains a k-mer for which the first $k-1$ nucleotides match with i and the last $k-1$ nucleotides match with j. The assembly is obtained by visiting every edge of a graph exactly once which is the Eulerian path problem. An example is illustrated in Figure 2.8.

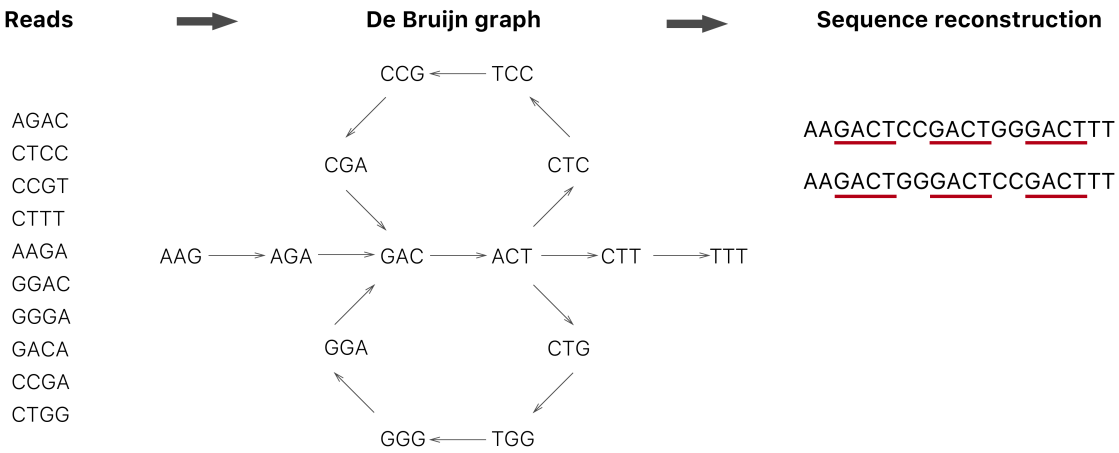

Figure 2.8: **De Bruijn graph of DNA sequence assembly.** All possible *k-mers* ($k = 3$ in this case) are created from a given set of reads. Then, a directed graph is built where nodes are *k-mers*. Two nodes are joined by a direct edge if the last two (k-1) bases of the first node match the first two bases in the second one. The sequence can be reconstructed by using the Eulerian path of edges.

The main advantages of choosing a de Bruijn graph for short reads data assembly rather than the OLC approach are the following:

1. There is no time and memory consuming calculations of overlaps between all reads. Instead, reads are processed to find all overlapping substrings of length k (k-mers). Each k-mer is stored once in hash tables. Therefore, de Bruijn graphs grow linearly with the input dataset size, which makes the assembly problem solvable for large genomes.

2. Since the k-mers are represented by edges, not nodes, the assembly is made using an Eulerian path of edges instead of searching for the Hamiltonian path

of nodes. Thus, the assembly can be found in polynomial time.

3. The de Bruijn graph approach originally was proposed to handle the assembly of repetitive regions which are hard to detect using the overlap-layout-consensus approach.

However, the de Bruijn graph approach is more sensitive to sequencing errors than OLC as it produces different k-mers. Therefore, the error detection step is very important. There are several types of sequencing errors that can be detected by assemblers: (i) base insertion, (ii) base deletion, and (iii) base replacement.

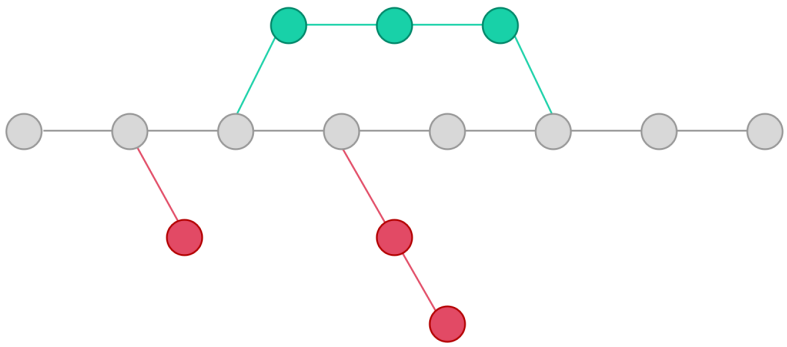

Figure 2.9: **Tips and bulges in de Bruijn graph.** The sequencing errors cause different structures forming in a graph, such as dead-end branches (tips) and cycles (bulges).

The errors lead to different structures forming in a graph that have to be resolved before finding the Eulerian path. There are three types of such structures: tips, bulges and whirls [64]. Tips are branches that end in a dead-end situation. They occur when sequencing errors happen at the end of the reads. Bulges are branches that create undirected cycles and are caused by substitution errors and indels usually in the middle of reads. Whirls are branches that create directed cycles and happen due to short tandem repeats. Bulges and whirls can be small, large or complex containing other bulges/whirls. An approach for removing bulges was proposed in [64].

Existing assemblers follow two approaches when dealing with errors: (i) error correction in reads [64, 14], and (ii) bulge/tips removal [64, 14]. However, the removal of bulges leads to errors in assembled contigs since important information may be lost.

2.4 Genetic polymorphism

To determine the genomic differences between individuals and to understand how they influence the phenotypic differences within a species are the main goals of genomics. The variations between human genomes consist of single nucleotide polymorphisms (SNPs), small insertions or deletions (indels), and structural variants (SVs).

A single nucleotide polymorphism is a variation like insertion, deletion or substitution in a single nucleotide, that occurs at a specific location in the genome. Currently, SNPs are the best studied and catalogued genetic variants.

Short indels are insertions and deletions that are no longer than 50 bp. They are easy to detect since the length of short reads produced by NGS technologies is long enough to cover a full indel.

Structural variants, on the other hand, can be defined as the genomic changes among individuals that are not single nucleotide variants, but large changes in genome structures (longer than 50 bp) [73]. These include large insertions and deletions, duplications, inversions, and translocations.

The single nucleotide polymorphisms and small insertions and deletions can be relatively easily identified and characterized by examining the mismatches and small gaps in their alignments with the reference genome. However, the detection of SVs, which involve sequence changes or rearrangements at a much larger scale, is much more difficult to detect due to the short length of sequencing reads provided by most NGS platforms [59].

Even though the new high throughput sequencing technologies and the whole genome sequencing method made it feasible to examine these variants, the characterization of all types of SVs is very challenging due to the ambiguous mapping of repeats.

The most challenging type to detect among SVs is insertions since they comprise a larger piece of sequence that are not in the reference sequences. Insertions can be further classified into (i) duplications - insertions of sequence also present elsewhere in the genome, e.g., mobile elements; and (ii) novel sequence insertions - insertions of a unique sequence that is not similar to other regions of the reference genome [34].

Currently, the actual number of structural variants, small or large, in an individual human genome is unknown.

Despite being lower in numbers, SVs contribute more to genome variants by sequence length, than SNPs or indels, and very likely impact more on gene functions. Therefore, analysis of SVs is a very important part of personal genome analysis.

2.4.1 Genome structural variants

Structural variants are common in human genomes. They constitute more than 50% of the genome and are the major cause of phenotypic variations [75, 21]. They are common in certain diseases, particularly cancer, and now they are showing up in complex diseases, such as autism and schizophrenia [69]. Furthermore, somatic structural variants play a central role in aggressive cancer development [48, 12]. It is critical to discover the precise mapping and sequences of SVs in order to associate them with phenotypes, and understand the effects on human diseases and evolution.

Structural variants vary widely in size and there are relatively many types of structural variants (Figure 2.10):

1. Copy number variations (CNVs) which include:
   (a) Insertions: a sequence of nucleotides added between two adjacent nucleotides in the sequence.
   (b) Deletions: a sequence of nucleotides was deleted.
2. Mobile elements insertions: a kind of insertion where the inserted sequence is a mobile element (highly repetitive sequence).
3. Inversions: a section of sequence occurring in an inverted orientation in relation to the reference.
4. Translocations: a region of nucleotide sequence that has translocated to a new position.
5. Tandem duplications: two adjacent segments are identical.
6. Interspersed duplications: nucleotide sequences that are nearly identical and can be found in multiple locations in the genome due to duplication events.

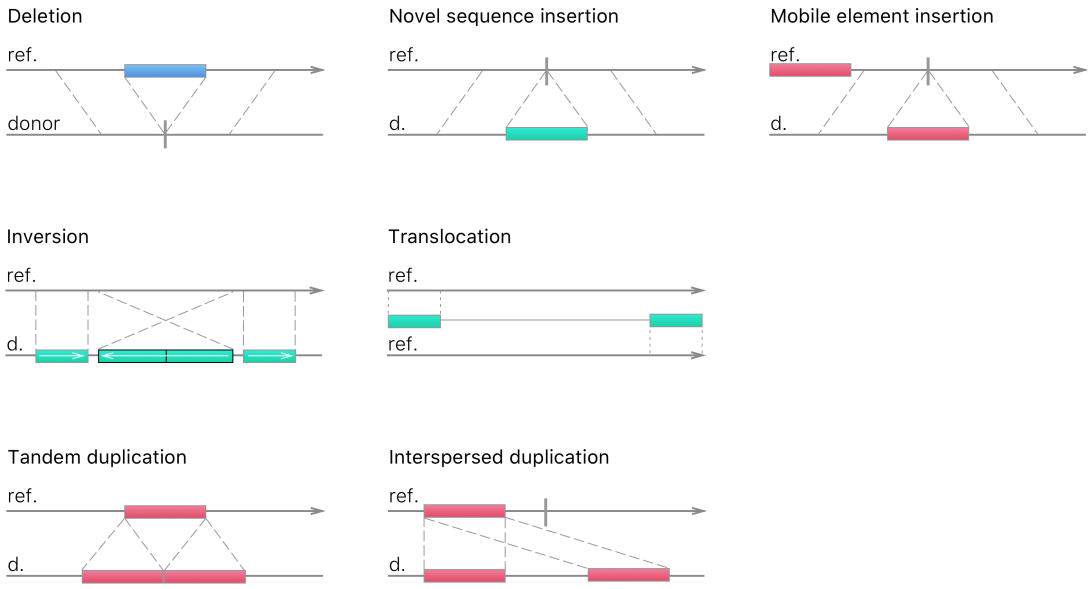

Figure 2.10: **Types of structural variation.** The figure depicts deletions, novel sequence insertions, mobile element insertions, inversions and translocations, tandem and interspersed duplications in a donor genome (bottom lines) when compared with the reference genome (upper lines). Image was adopted from [6].

2.4.2 Detecting structural variants

To compare individuals' genomes, current methods use a donor sequenced genome and the assembled reference genome. Since it is impossible to directly compare the sequences, they rely on signature patterns of paired-end reads that are made by structural variants [55].

2.4.2.1 SVs signatures

The signatures that are used to detect SVs can be divided into basic ones for identifying approximate locations of breakpoints and ones with precise breakpoint identification. Further, these signatures are subcategorized according to the type of SV they detect.

1. **Basic**

   (a) Insertions and Deletions. For an insertion event, the mapping distance to the reference genome is shorter than the insert size (Figure 2.11:a). For a

deletion event, the corresponding distance is longer (Figure 2.11:b). When the size of the insertion is bigger than the insert size of sequenced reads, the basic insertion signature will not be detected. Moreover, the insertion signature does not identify the inserted sequence.

(b) Inversion. Paired-end reads that span one of its breakpoints is mapped to the reference with one of the reads in an opposite direction (Figure 2.11:c).

(c) Linking insertion. Two adjacent regions in a donor genome appeared at a distance in the reference (Figure 2.11:d). In the linking insertion signature, read mates that span the breakpoint in a donor genome will be mapped to the reference with a distance much greater than the insert size. However, for a very large insertion, this signature is weak because there is no confidence that two linking signatures indicate the same insertion.

(d) Everted duplication. A mate pair that has an end in each of the two copies will have an everted mapping: the order of the mates is reversed while the orientation stays the same (Figure 2.11:e). It can only be used to detect a novel tandem duplication (a duplication consisting of two identical adjacent regions).

2. **Breakpoint identification**

(a) Split mapping. One of the mates is fully mapped to the reference. The other read from a pair is split into prefix and suffix that are mapped to different locations in the reference. For a deletion event, the suffix and prefix will be split apart (Figure 2.11:f) when mapped to the reference, while for an insertion they will overlap (Figure 2.11:g). This signature can pinpoint the precise location of an event.

(b) Hanging insertion. One of the read mates is mapped to the reference and the other one is not mapped (Figure 2.11:h). This signature can be used to detect short insertions since for long insertions, hanging reads will not cover it entirely.

3. **Depth of coverage (DOC) signature**

Assuming the sequencing process is uniform, the number of reads mapping to a region follows a Poisson distribution and is expected to be proportional to the number of times the region appears in the donor. Thus, a region that has been deleted (duplicated) will have less (more) reads mapping to it. This signature

does not indicate exactly where an insertion occurred, but rather the relative copy number of a region of sequences present in the reference. Thus, it is not able to detect insertions of novel sequences.

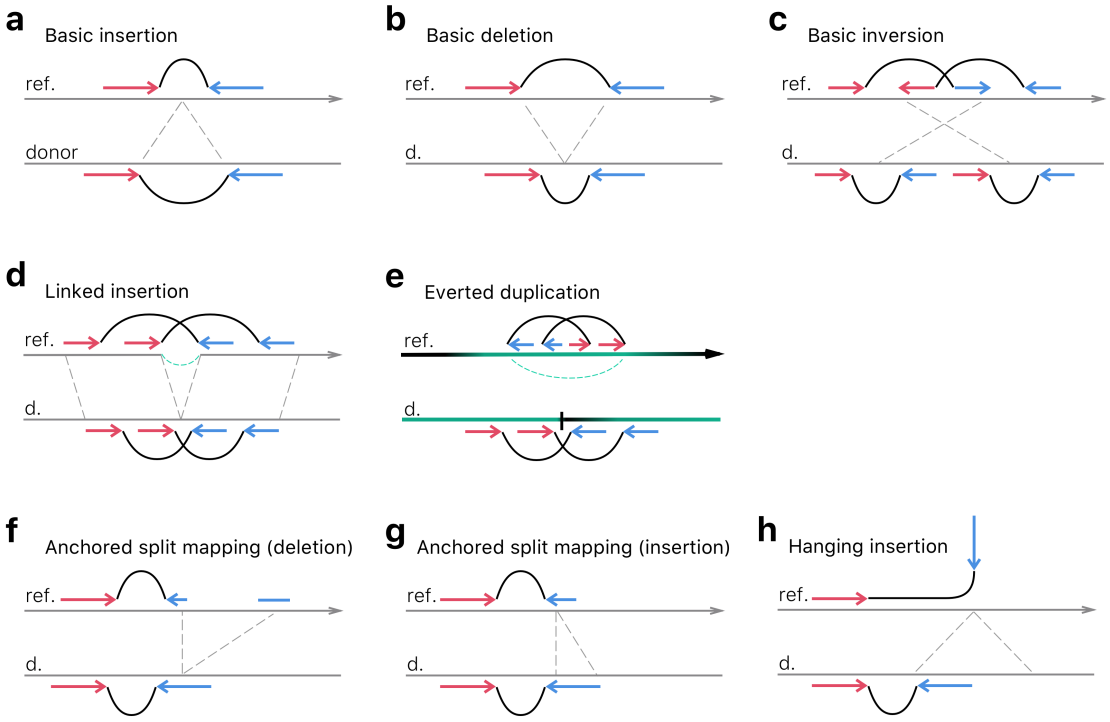

Figure 2.11: **Structural variants signatures.** The figure depicts signatures that are left by SVs in a genome and can be detected by examining the mapping patterns of paired-end reads. Basic signatures include (a) insertions and (b) deletions, when the mapping distance differs from the insert size, and (c) inversions, where the orientation of mates is changed. (d) A linked insertion occurs when the sequence is copied from another location in the genome. (e) When mates are in proper orientations but the order is abnormal, an everted duplication signature can be observed. This signature is created by a tandem duplication. (f,g) An anchored split mapping signature is observed when one mate is fully mapped to the reference, whereas the other has a split mapping. (h) When one mate is mapped and the other is unaligned, we have a hanging insertion signature that indicates a novel sequence insertion. Image was adopted from [55].

2.4.2.2 SVs detection methods

The first methods used for CNV discovery with Sanger data used hybridization-based microarrays. There are two basic microarrays representations: the SNP microarrays

[52] and the array comparative genomic hybridization (array CGH) [26]. Both of them conclude copy number gains or losses compared to the reference, but they differ in implementation details [6].

Microarrays advanced in throughput and cost, and they are able to assay CNVs of large data sets. However, they cannot identify the locations of duplicated sequences and provide breakpoints at the single-base-pair level.

With the advent of next-generation sequencing technologies, microarrays were replaced by new methods that focus on searching discordant mapping patterns for structural variants discovery.

There are four basic sequence-based approaches for SV detection which can discover variants of all types, but each biases to a different type of SV due to the properties of sequence reads. Figure 2.12 illustrates these four types.

1. *Paired-end (or read-pair) mapping*: assesses the abnormal mapping distance and orientation of paired-end reads and can detect highly repetitive CNVs and define their locations. This can detect insertions, deletions, and inversions as well. However, it relies on a confident independent mapping of each end (problems in regions flanked by repeats), and the resolution of complex regions or breakpoints depends on the span (interval) between ends. The accuracy and sensitivity depend on coverage, read length and insert size. Among the most well-known tools that are based on a read-pair approach are VariationHunter [22, 23, 24] and BreakDancer [15].

2. *Read depth analysis*: assumes a random (generally Poisson) distribution in mapping depth and examines the increase and decrease in coverage to detect duplications and deletions respectively. However, it has difficulties in distinguishing highly repetitive CNVs and provides no precise locations of the breakpoints. Chiang *et al.* [16] and Abyzov *et al.* [4] attempted to discover deletions and duplications at better breakpoint resolution using the read depth-of-coverage.

3. *Local assembly* of sequence not present in the reference genome. It is able to detect SVs of all types at the breakpoint resolution, even better than all other methods. But in highly repetitive or duplication-rich regions the produced contigs/scaffolds are fragmented. Some of the *de novo* assembly algorithms for NGS whole-genome shotgun data are EULER [13], ABySS [68], SOAPdenovo [44], and NovelSeq [20].

4. *Split-reads analysis*: is able to detect precise breakpoints of all variant types by

analyzing the alignment to the reference. Compared to other methods, split-reads analysis is weaker in repetitive and duplication-rich regions. This approach makes it possible to detect mobile element insertions (MEIs) if reads are long enough to span the event in order to characterize the sequence content. In other cases, the split-reads method will be able to anchor the insertion. Due to the difficulty in aligning short reads there are few algorithms that use split-reads for SV detection. Among them are AGE tool [3] and Pindel algorithm [79].

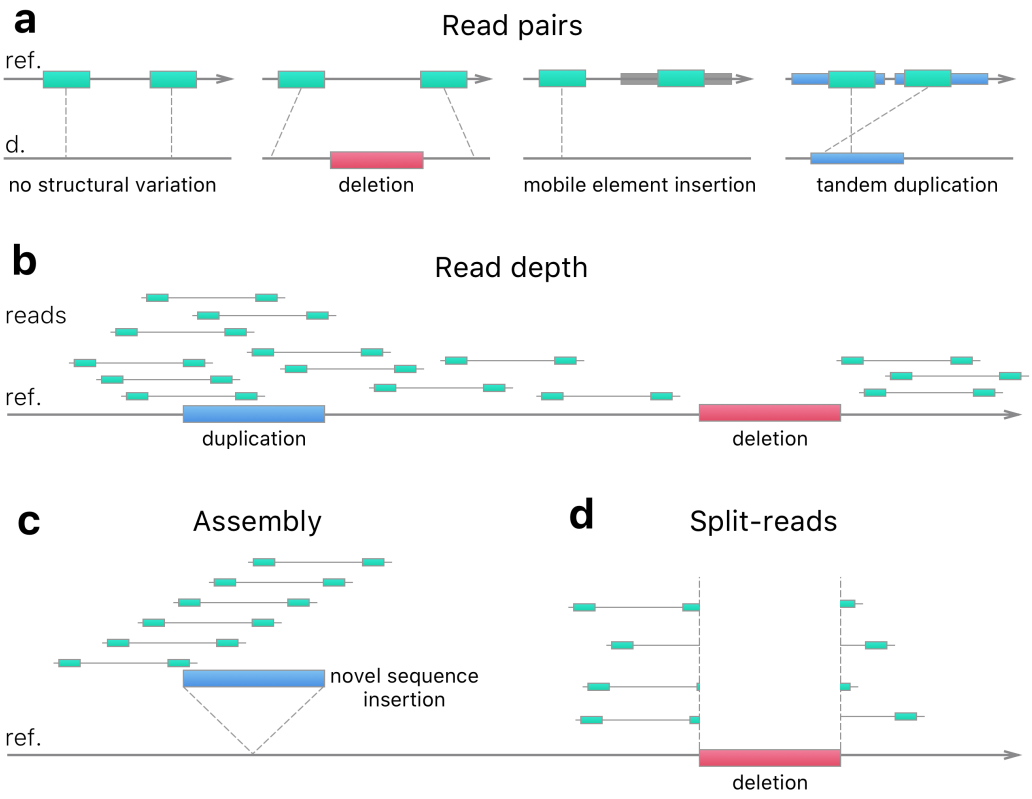

Figure 2.12: **Methods for structural variants discovery.** (a) The read pairs (or paired-end) mapping method examines the mapping distance and orientation of paired-end reads to detect insertions, deletions, and inversions. (b) A read depth analysis assesses the coverage distribution to detect duplications and deletions. (c) A local assembly can characterize a novel sequence insertion. (d) A split-reads analysis is able to detect all types of SVs and provide the precise breakpoints.

Currently, all computational and experimental methods for SV discovery generally focus on one type of SV due to the different frequency and size range of different types.

Insertions remain one of the most challenging types of variation to detect because (i) their detection from short read sequencing data is challenging; (ii) insertion detec-

tion requires a *de novo* assembly; and (iii) typical genotype callers use only reference-aligned read pairs and, therefore, are not suitable for calling insertions longer than the read. SVs often involve repeated regions and complex rearrangements which complicate their precise detection.

2.4.3 Mobile elements insertions

Mobile elements (or transposable elements) are highly repetitive DNA sequences that are copied and moved through the genome. They are the most common SV type by number. Mobile elements constitute nearly half of the human genome as a result of repeated insertion events during genome evolution, and in some plants, they appear in up to 90% of the genome [31].

Mobile elements are the major cause of genomic architecture changes through evolution. It was found that MEs have a mutually beneficial relationship with genes by providing their sequences for protein-coding exons of genes and affecting gene expression [31]. Moreover, they can disrupt gene functions by inserting into functionally important regions, thus provoking a number of human diseases.

Although most of the transposable elements are now fixed in the population, some MEs are still actively duplicating [78]. Mobile element insertions (MEIs) have been associated with human genetic disorders, including Crohn's disease [53], hemophilia [32], and various types of cancer [56, 39], motivating the need for accurate MEI detection methods.

It was found that there are over 90 disease-producing MEIs, including of 60 Alu elements, 25 L1s, and 7 SVAs [72].

Depending on the means of transposition, mobile elements are divided into two classes, namely DNA transposons and retrotransposons, which can be further subdivided into autonomous retrotransposons, and non-autonomous retrotransposons [78].

1. *DNA Transposons.* These elements are excised from one genomic site and integrated into another by a cut and paste mechanism.

2. *Retrotransposons.* Retrotransposons are moved by a copy and paste mechanism using reverse transcriptase (RNA to DNA). First, they are transcribed into RNA, then reversed transcribed, and then reintegrated into the genome. Thus, the element is duplicated. The subclasses of retrotransposons either contain long terminal repeats at both ends (LTR retrotransposons) or lack LTRs (non-LTR retrotransposons) [31].

Non-LTR retrotransposons mobilize via RNA intermediates using a mechanism called target site-primed reverse transcription (TPRT). In the TPRT process, an RNA copy is first generated from the original retrotransposon and subsequently reverse-transcribed back into DNA and then inserted into the genome. During the process, two short stretches of identical sequence, called target site duplications (TSDs), are created on both ends of the new insertion. This is illustrated in Figure 2.13.

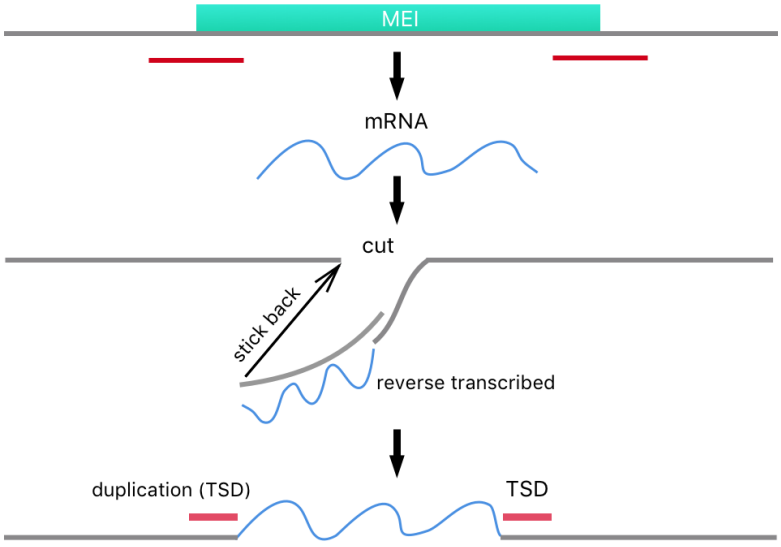

Figure 2.13: **Formation of target site duplications.** In the beginning, an RNA copy is created. Then, it is reverse-transcribed into DNA and inserted back into the genome. After the insertion, two short identical sequences are generated on both ends. These sequences are called target site duplications.

Together DNA transposons and retrotransposons compose around 45% to 69% of the human genome [72, 77] and they have a great impact on genes: (i) they influence mutations in genes, (ii) they can affect gene expression, (iii) they can move genetic elements, etc. [9].

Only retrotransposons remain active in the human genome and contribute to variation between individuals in the population. Therefore, in our work, we consider only retrotransposons.

There are four types of insertional mobile elements in the human genome:

1. **Alu**

   The Alu family is the most common ME in human genomes. They have expanded to 1.1 million copies of up to 300 bp in length each, which together

constitute up to 11% of the human genome [70].

According to sequence alterations, Alu elements were further classified into subfamilies. The most currently active Alu subfamilies are AluY subfamilies (e.g., AluYa5 and AluYb8 [70]). The reference genome contains over 140,000 annotated AluY elements.

Alu events occur in at least 1 in every 30 individuals and have caused over 20 human genetic diseases. Moreover, Alu elements increase genetic diversity [31].

The typical Alu structure is shown in Figure 2.14. The Alu sequence is flanked by two target site duplication (TSD) sequences. The poly A tail (a number of As) appears at the end of the insertion if it is mapped to the "+" strand. Otherwise, a poly T tail will be present at the beginning of the insertion sequence.

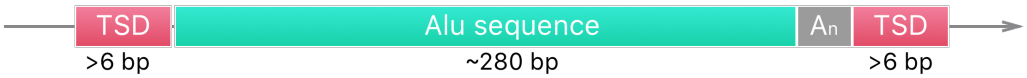

Figure 2.14: **The structure of an Alu element.** An Alu sequence is up to 300 bp long. The insertion part is followed by a poly A tail and is flanked by TSD.

2. **SVA**

SVA elements are nonautonomous, non-LTR retrotransposons composite mobile elements named after its main components, SINE (short interspersed element), VNTR (variable number of tandem repeat) and Alu. SVAs range in size up to 3 kb and have more than 3,600 annotated elements in the reference genome. SVA elements are considered to be the youngest family of non-LTR retrotransposons [70] and occasionally generate disease-causing insertions in humans. There are six subfamilies of SVA that were named SVA_A to SVA_F.

The structure of a full-length SVA element is presented in Figure 2.15. Most SVAs are flanked by TSDs. The 5' end contains a variable number of hexameric (CCCTCT) repeats, followed by an antisense Alu sequence, a VNTR region containing multiple copies of a 35-50 bp repeat, an SINE-R region, and a poly A signal (AATAAA) that is followed immediately by a poly A tail [62].

The SINE-R element is a novel retrotransposon which is derived from a human endogenous retrovirus, HERV-K10 [60].

Figure 2.15: **The structure of an SVA element.**

3. **L1**

   Long Interspersed Elements (LINE-1s or L1s) can be up to 6 kb each. There are over 500,000 L1 elements which make up 17% of the human genome. L1HS is the most common subfamily that has more than 1,500 elements annotated in the reference genome [70].

   L1 elements affect the genome in both destructive and constructive ways. Insertion and rearrangement are destructive processes. However, L1s occasionally repair double-strand breaks in DNA [31].

4. **LTR**

   Long terminal repeat (LTR) retrotransposons can be up to 10 kb in length and have a significant impact on genome structure and functions. The structure of an LTR retrotransposon is represented in Figure 2.16. There are two sites, the primer binding site (PBS) and polypurine tract (PPT), that are critical to replication. Every LTR is surrounded by long terminal repeats (LTRs) which typically end in the dinucleotides TG and CA. During the self-insertion into host DNA, a short segment of the host DNA, called target site repeat or TSR, is replicated at the ends of insertion [54]. The internal region contains the group specific antigen (gag) gene, the polymerase (pol) gene, and envelope (env) gene.

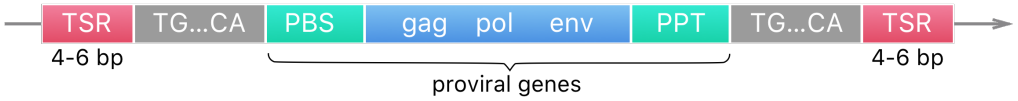

Figure 2.16: **The structure of an LTR element.**

Mobile elements can also be divided into reference and non-reference ones depending on whether the insertions are present in the reference genome. However, there are only a small number non-reference MEIs with full sequence characterization.

Over millions of years of evolution, a balance between damaging effects on an individual and long-term beneficial effects on a species of genome modifications was achieved. Probably, soon it will be proven that mobile elements played an important role in speciation by shaping the genome [31].

2.5 Tools for detecting structural variants

In recent years, many computational methods were developed to detect and characterize structural variations in the human genome using NGS platforms. However, the characterization of SV sequences, especially longer ones, remains uncovered due to the short insert size of read libraries produced by the NGS method and the high repetitiveness of SVs. Moreover, sequencing errors and genome diversity complicates the process of assembly for long insertions.

Several tools have been created to detect structural variants but only few of them can cover mobile element insertions. Among them are: Tangram [77], PopIns [34], PopAlu [66], RetroSeq [33], and Mobster [72]. These methods focus mainly on pinpointing breakpoints of SV or on the detection of novel insertions, and generally follow a three-step approach:

1. First, they identify fragments (reads or read pairs) that indicate the occurrence of an SV.

2. Next, they cluster these fragments along the genome, such that each cluster represents a potential SV event.

3. Last, for each sequenced genome and at each cluster, they calculate a likelihood that an event has actually occurred given the set of fragments, and breakpoints are derived.

But unlike novel sequences, MEIs are repetitive, i.e. are inserted at more than a single location in the genome, which is why the novel sequence discovery programs do not detect mobile element polymorphisms.

There is one more tool that can detect SVs and provide sequence characterization for short insertions - the DELLY tool. DELLY [67] is an open-source software for identification of deletions, insertions, inversions, translocations, and duplications. For deletions, it covers reference MEI deletions and in most cases, it can provide the full sequence representing the pre-integration allele (i.e. the allele without the MEI). Unfortunately, the length of insertions it can detect and characterize is in the range

of 15 - 120 bp (maximum). Large insertions (such as MEIs) with lengths of 300 - 10 kb remain uncovered.

2.5.1 Tangram - MEI detection toolbox

Wu *et al.* developed the Tangram tool [77] to target the MEI detection problem. The novelty of their work was the integration of both read-pair and split-read mapping signals. They performed a split-read mapping step before the beginning of the detection, and, as a result, these mappings could nucleate SV event calls. This approach also improved the accuracy of SVs' breakpoints identification. Tangram targets both LTR and non-LTR mobile elements and provides genotype likelihoods as well. The main features that made Tangram stand out among other detection tools were the ability (i) to simultaneously process multiple sequence alignment files and (ii) to deal with multiple fragment length libraries and a mixture of read lengths within a single detection step.

Tangram was run on 1000 Genomes Project data, and according to the reported results, it was able to pinpoint MEI breakpoints with single-nucleotide precision [77].

Overall approach. There are three modules that are integrated into a pipeline: RP (read-pair), SR (split-read) and genotyping.

The input is represented by two kinds of reads: the ones that are aligned to the reference genome and the ones that are aligned to ME reference sequences. These reads are stored in customized binary alignment (BAM) files that have an extra ZA tag indicating that a mate maps to one of the ME reference sequences. The alignment part is conducted using the MOSAIK mapping software. The MOSAIK program uses hashes to prioritize reads that are mapped to the ME reference sequences to the reads that are mapped to the genome reference. Potentially, there can be hundreds of mapping locations for MEs since they are very repetitive. After the MOSAIK alignment, the ZA tag is added to the BAM file containing information about the MEI location, mapping quality and information about the read mate and the number of its mapping locations. The ZA tag helps to search through the BAM file faster.

Firstly, the PR submodule searches the read pairs where one mate is uniquely aligned to the genome reference, while the other one is aligned to an ME reference. See the red arrows at the top part of Figure 2.17. These read pairs must satisfy the following requirements: (i) the orientation is different from expected; (ii) reads that are mapped to the reference genome are aligned to different chromosomes; and (iii)

the length of the fragment does not follow the fragment length distribution calculated at the beginning using concordant reads (read pairs where both reads are aligned to the same chromosome at the expected distance and orientation). Then, using a customized nearest-neighbour algorithm, the module clusters uniquely mapped mates for each ME type the way that read pairs clusters are within a range determined by the fragment length distribution. 5' end read pairs are clustered separately from 3' end. In Figure 2.17 the green dashed boxes represent the clusters. An MEI event is identified if the pair of neighbour clusters span into the insertion from both ends (5' and 3'), having the breakpoint as the smallest coordinate between the end of the 5' cluster and the beginning of the 3' cluster.

Secondly, read pairs where one mate is fully aligned to the genome reference, and the other mate is either soft-clipped or unaligned are collected and passed to the SR mapping submodule (Figure 2.17 red arrows at the bottom part). The SR submodule integrated the Scissors package to run a split-read algorithm that aligns one part of a read to the reference genome and another part to the ME reference. Thirdly, loci with a potential MEI event are extracted based on RP or SR evidence and filtered by the number of supporting fragments. Candidates with at least two RP supporting fragments from both ends or two SR supporting fragments are kept.

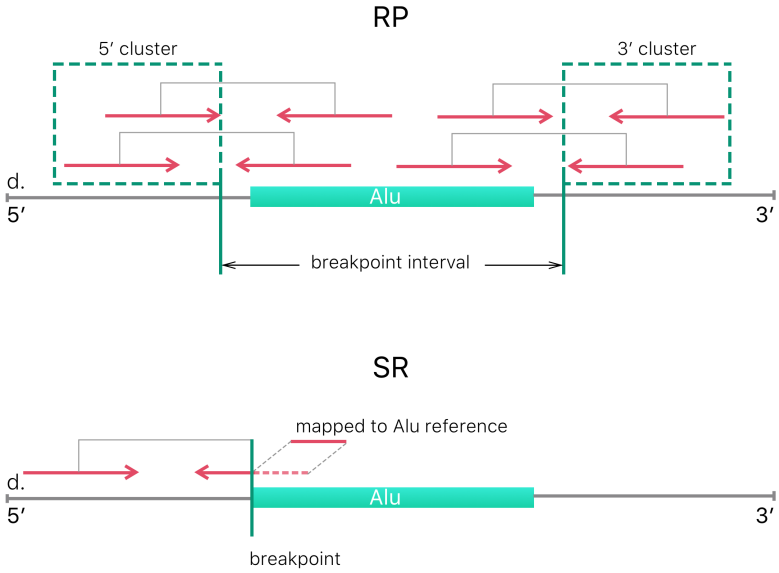

Figure 2.17: **Illustration of RP and SR ME detection approach.** The top part illustrates the RP phase. A gray line represents a test genome with an MEI. A read pair is represented by red arrows. During the RP phase, the pairs where one mate is mapped to the genome and the other is mapped to the MEI are collected and clustered (green dashed boxes). The location is estimated based on the reads' locations in clusters. The bottom part illustrates the SR method, where the split-reads are collected (one mate is mapped to the genome, whereas the other is either unaligned or soft-clipped). One part of the soft-clipped read is mapped to the genome and the other is aligned to the ME reference. The alignment location of the first segment of soft-clipped read determines the breakpoint. Image was adopted from [77].

After that, genotype likelihoods are calculated using the Bayesian framework. Finally, a report is produced in the VCF format that includes the location, the type, and the individual sample genotype information about each event.

Compared to other MEI detection tools, Tangram showed superior sensitivity, breakpoint resolution, and genotyping accuracy. However, the tool does not provide the MEI sequence characterization.

2.5.2 PopIns: novel insertions characterization tool

The PopIns tool [34] was created to address the problem of long novel sequence insertions detection and characterization. The novel insertions are challenging to characterize since they do not have similarity in the reference genome and short read

data requires *de novo* assembly. For the *de novo* assembly high-quality data is required, which is difficult to obtain from a single individual. Therefore, PopIns utilizes reads on a population scale. As a result, PopIns is able to detect and characterize novel insertions of 100 bp and longer and provide the genotype information for all individuals in the set.

The approach is based on a simultaneous characterization of insertions across a large number of individuals using a local assembly strategy.

Firstly, they defined three subproblems that should be solved for a single individual:

1. *Assembly.* Reads that are not aligned to the reference genome are assembled into high-quality contigs. The Velvet tool was integrated to cover the assembly part.

2. *Positioning.* The positions of contigs in the reference genome are determined using read-pair and split-read mapping information. Firstly, the probable location is found using anchoring reads, then split-reads are used to get the exact insertion location. Although there can be multiple possible positions, only one is chosen. The locations with high anchoring score and of length shorter than twice the maximum allowed insert size are kept. Then for each location, the exact insertion position is determined using split-reads.

3. *Genotyping.* The genotype information is determined for each individual - how many copies of an insertion it carries. Possible genotypes are the homozygous reference (zero copies), heterozygous (one copy), or homozygous insertion (two copies).

Subsequently, this single-individual approach is extended to cover the set of data of multiple individuals by introducing the *merging* step. The merging step is added right after the assembly subproblem in order to benefit from multi-individual data. The set of contigs that are produced in the assembly step is merged into a single supercontig that meets the following requirements: (i) any two supercontigs cannot be merged further, and (ii) every contig is represented by a single supercontig.

The merging step is the major novelty of the PopIns approach, and this allowed the researchers to characterize long novel sequence insertions. However, non-unique sequences (e.g., MEI) cannot be covered by this tool since the contigs will not be assembled from unaligned reads.

2.5.3 RetroSeq: transposable element discovery

RetroSeq [33] is a software for detecting non-reference transposable (mobile) elements (TE) insertions with high accuracy.

As an input, RetroSeq requires a BAM file with both aligned read pairs and pairs with one mate unaligned, the reference genome, and a dataset of known TE elements locations or ME sequences.

There are only two phases implemented in the RetroSeq approach:

1. *Discovering.* Discordant read pairs are collected and assigned to a class based on their type (Alu, SVA, L1, etc.). The type information is available either from a dataset of all annotated TE in the reference or from the alignment to the library of ME sequences.

2. *Calling.* At this phase, the anchoring reads of TE candidates are clustered based on their location and the strand to which they are aligned. Then, these forward- and reverse-strand clusters are merged into regions around supported breakpoint. Using soft-clipped reads the exact breakpoint is determined.

RetroSeq showed the same sensitivity as the Tangram tool ($> 97\%$) for Alu and L1 detection in three individuals from the 1000 Genomes Project.

2.5.4 Mobster: detection of MEI in NGS data

More recently, Thung *et al.* presented a novel method for active non-reference MEIs detection called Mobster [72]. The algorithm works with both whole-genome (WGS) and whole exome sequencing (WES) data. The difference between these two data sets is implied in the naming: WGS attempts to sequence the entire genome and it can cover only 95-98% of the genome, while WES focuses on protein coding sequences.

Mobster is able to detect all active families of MEIs. As an input, Mobster takes a binary alignment (BAM) file and extracts all discordant and clipped reads that are used to pinpoint a potential non-reference MEI event.

The criteria for discordant read pairs are: (i) reads are aligned to different chromosomes, (ii) the distance between aligned reads significantly differs from the median insert size, (iii) the orientation of aligned reads is different from expected, and (iv) only one read is aligned, while the other one is not. To increase sensitivity and specificity of SVs detection, Mobster uses all discordant read pairs, not just the ones with multiple alignments.

The read from the anchoring read-pair that is aligned to the reference genome is used to determine the breakpoint of a possible event, while the unaligned read is mapped to the custom library of known active ME consensus sequences, called mobilome. The clipped sequences of clipped read-pairs are also aligned against the mobilome and investigated for having a poly A/T tail. After that, they are tagged according to the MEI type to which they were mapped (Alu, SVA, L1, or HERV-K).

Anchors of discordant and clipped reads are clustered separately. For clipped clusters, reads should satisfy the following requirements: (i) they should support the same MEI family, (ii) they are clipped on the same side, and (iii) they are clipped within couple base pairs of each other.

Clipped clusters from both sides (left 5' end and right 3' end) indicate the same MEI if: (i) they carry the same ME type or one of them supports ME type and the other carries a poly T/A tail, and (ii) they either overlap (maximum 50 bp for TSD) or are separated (maximum 20 bp - TSD deletion).

Discordant clusters satisfy the conditions: (i) clusters are mapped to the same strand, (ii) they support the same ME type, (iii) they have a starting position within a specified distance. Discordant clusters from 5' end and 3' end indicate the same MEI when they overlap (maximum 50 bp) or are within a user-specified region.

Mobster was able to detect 4-5% of novel MEIs in comparison to the MEI reference set provided by dbRIP. However, the tool is not able to discover the actual sequence of detected MEIs.

2.5.5 DELLY: SV discovery tool

The DELLY [67] tool combines paired-end mapping (short- and long-range) and split-read analysis for structural variants discovery at single-nucleotide resolution. They achieved high sensitivity and specificity in detecting deletions, inversions, tandem duplications, and translocations. The distinctive feature is that DELLY can process different sequencing libraries with different insert sizes.

The input files are in SAM/BAM format containing aligned reads. Each file is a separate library with a distinct insert size median and standard deviation. To achieve optimal sensitivity, all input files are analyzed jointly.

The main approach combines two separate components: (i) paired-end mapping, and (ii) split-read analysis.

Paired-end mapping. For each input file, the default read pair orientation and the paired-end insert size are calculated. Then, discordant read pairs that violate

either the orientation or insert size are collected.

DELLY leverages the following patterns to characterize SV types:

1. *Deletions*: paired-end reads that have default orientation but the insert size values are at the end of the insert size distribution.

2. *Inversions*: paired-end reads with abnormal orientation.

3. *Tandem duplications*: paired-end reads where the reads changed their order but the orientation is kept as the default.

4. *Translocations*: paired-end reads that are mapped to different chromosomes.

All discordant paired-reads are sorted according to the left-most position including chromosome information. This sorted vector is used to build an undirected, weighted graph that indicates which read pairs support the same SV. A node in the graph represents a read-pair and the edge denotes that connected read pairs support the same SV. Paired-end reads with the same mapping pattern are clustered together to achieve maximum specificity.

Ideally, the graph will contain one fully connected component for each SV. Thus, each variant could be defined by computing the connected components of the graph. Unfortunately, most components are not fully connected due to sequencing errors, incomplete reference sequences, and ambiguous mapping locations. Therefore, the maximum clique is used to estimate the start and end positions of a structural variant.

Split-read analysis. The paired-end clusters are considered to contain the breakpoint interval, and they are screened for split-reads to determine the genomic variation at single-nucleotide resolution. The split-read analysis is conducted in several steps:

1. Searching for split-read candidates. For each SV interval, DELLY searches for single-anchored paired-ends - a read pair where one read is mapped to the reference and the other one is not mapped. The unmapped read potentially can be a split-read. Then, the split-reads are collected according to the following conditions: (i) all SV start and end breakpoints are sorted by chromosome and position, (ii) BAM files are screened for single-anchored reads, (iii) using the read's mate and default orientation of the library, determine the search direction, (iv) find the closest SV breakpoint using binary search, and (v) if a read aligns within two standard deviations of a breakpoint, assign this read as a split-read candidate.

2. Extracting the SV reference. The paired-end SV interval is extracted from the genome. Depending on SV type, the reference is modified in such a way that the prefix and suffix alignments of a split-read are in the same orientation and the expected order. For example, the orientation is changed for inversions, the prefix-suffix order is changed for tandem duplications, and they are both changed for translocations.

3. Indexing and counting of k-mers. To identify candidate split-reads efficiently, DELLY uses a k-mer-based filtering technique. The default value of k-mer-index is 7. This k-mer-index contains the SV reference for the given SV region. DELLY aligns each k-mer of a read from a given SV region against the index and hits are counted by alignment diagonal. If after the post-processing a read has less than two diagonals above the defined threshold, it is discarded from the set of putative split-reads.

4. Detecting the breakpoint.

5. Computing the consensus of the split-reads for each subset of reads that support the breakpoint offset. The majority vote is applied in each consensus column.

6. Aligning split-read consensus to the SV reference region. DELLY resembles an AGE algorithm [3] developed for contig alignment to the reference genome. However, the researchers made a couple of changes: (i) gap penalties, and (ii) global alignment instead of local since the reads used to build the consensus should cover the full length.

The results showed excellent sensitivity by paired-end mapping and great specificity by split-read analysis while integrated together, otherwise, they are not optimal across all sequencing parameters. However, the DELLY tool is good for detecting reference polymorphic MEIs, but not non-reference MEIs.

There are many tools for detecting MEIs, but none provide full insertion sequence characterization.

Chapter 3

Approach

The primary objective of this work is to develop a tool for the sequence characterization of insertional structural variants in the human genome to address the limitations of the current human reference genome sequence and variant detection approaches. Full sequence characterization is achieved by utilizing all personal genome data available in the public domain.

A future goal is to build a pipeline that integrates these resources to streamline personal genome analysis and to construct a new database of structural variants that will contain the full sequences of all known MEIs.

All available genome sequence data covering different sequencing platforms are pooled together to achieve significant improvements in SV detection and sequence characterization. This approach gives us the ability to detect structural variants with a better accuracy and to provide full sequences or at least critical breakpoint sequences for the majority of non-reference SVs.

In this work, we focus on mobile element insertions (MEIs) because they are the most challenging due to high sequence repetitiveness. The proposed approach can be used in future works to analyze all the main types of SVs including non-MEI large insertions/deletions/CNVs, inversions, translocations, and tandem duplications, with slight modifications in sequence patterns and validations.

For reference MEIs, the full insertion sequences are available from the reference genome, and the target site duplication (TSD) sequences are deductible.

For non-reference MEIs, the database of Retrotransposon Insertion Polymorphism in Humans (dbRIP) [75] contains the full sequences for a total of 1,087 locations including Alu, SVA, L1, and LTR. This dataset was used for testing purposes.

In summary, we performed sequence characterization for a total of 1,613 polymorphic MEIs from 1kGP data that comprise 1,372 Alus, 95 SVAs, and 146 L1s.

3.1 Overview of the strategy

The goal of the tool is the sequence characterization of known SVs. We are using both discordant read pairs and split-reads to cover the breakpoint sequences and characterize the insertion. A read pair is considered to be discordant when at least one of the following conditions is true: (i) the mapped orientation is abnormal, (ii) the distance between mapped reads is different from expected, (iii) mates are mapped to different chromosomes, or (iv) one mate is mapped and the other mate is not mapped. This definition was proposed by Tuzun *et al.* [73].

A read pair is called concordant when two mates are mapped on the same chromosome to the opposite strands within the expected distance. In the Illumina platform, the orientation of a paired-end read is "correct" if the left mate is mapped to the "+" strand, and the right mate is mapped to the "-" strand.

Split-reads are reads that are split into two parts that are aligned to two distinct places of the genome with little or no overlap. They are indicative of structural variations.

We determine the breakpoints of an occurred SV as well as short sequences at the start and the end of the insertion by using split-reads and discordant reads. This information gives us a clue about the type of MEI and a particular MEI sequence that appears somewhere else in the genome. This MEI will serve as a template for our assembly.

Figure 3.1 represents the flowchart of our algorithm. It includes several phases that can be described in the following ways: (i) the data processing phase, where required discordant and split-reads are collected and validated, (ii) the "bridge" assembly, where the reads are assembled into contigs and then extended to cover an insertion sequence, (iii) extraction of the MEI information, and (iv) validation.

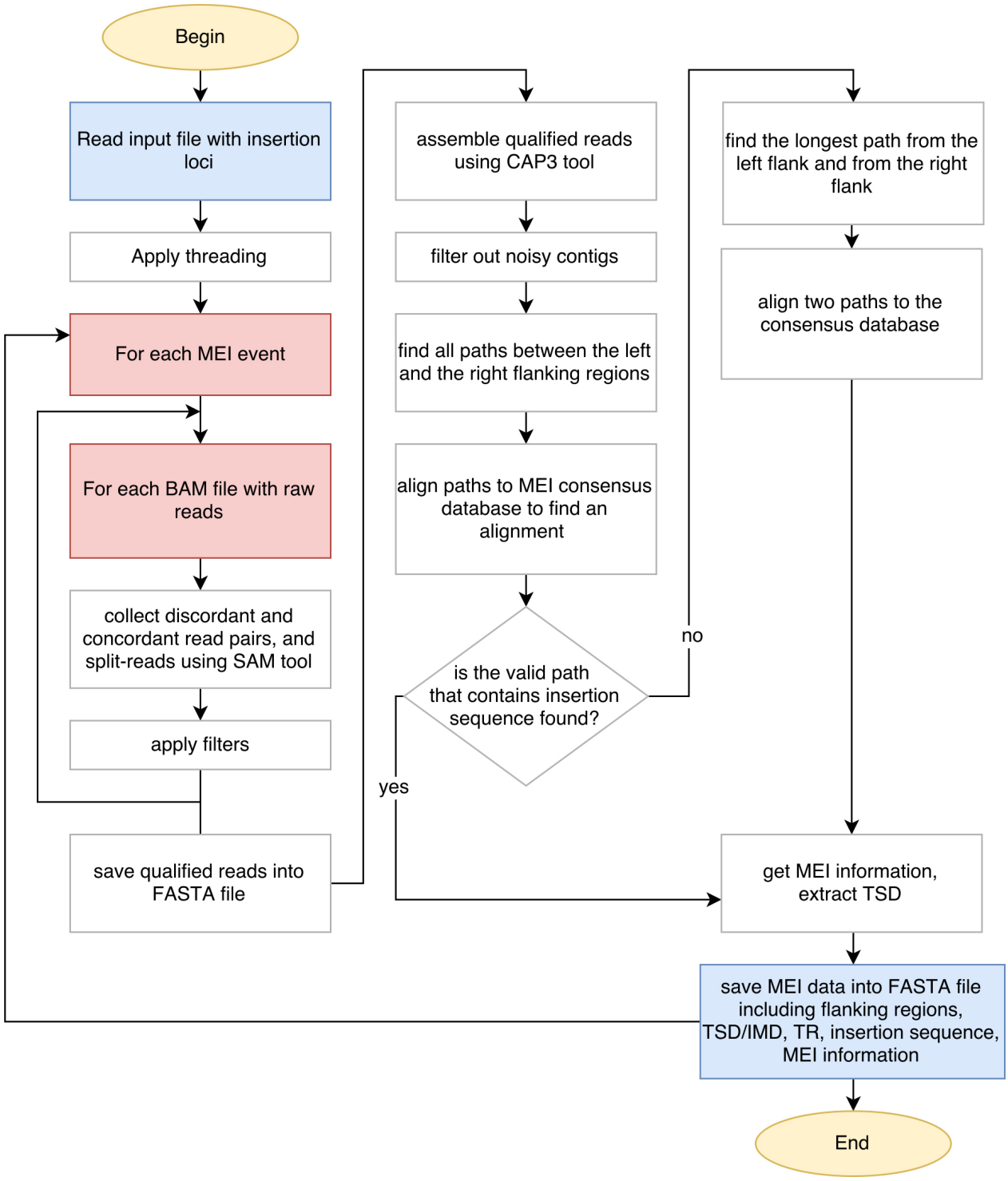

Figure 3.1: **Flowchart of the approach.**

3.2 Data processing

The non-redundant list of non-reference MEIs is provided in a Variant Call Format (VCF) or in a Browser Extensible Data (BED) format and contains the following information about the insertion: an approximate location (within 1-2 bp) in the reference genome, type if known, strand, and sample data. The examples of these formats are represented in Figures 3.2 and 3.3. We run our pipeline for each given loci.

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT          Alu.ASW Alu.BEB Alu.CDS Alu.CEU
Alu.CHS Alu.CLM Alu.ESN Alu.FIN Alu.GBR Alu.GIH Alu.GWD Alu.IBS Alu.ITU Alu.JPT Alu.KHV Alu.LWK Alu.MSL Alu.MXL
Alu.PEL Alu.PJL Alu.PUR Alu.STU Alu.TSI Alu.YRI
chr1    10014903        .       .       <INS:ME>        13      .       MEINFO=Alu,10014903,10014894,NA;NOT_VALI
DATED;SVTYPE=INS        GT:GQ:FL:SP:CN  0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/1:13:8
:46:2   0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:
0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:
0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0       0/0:0:0:0
```

Figure 3.2: **Excerpt of a VCF file.** The header line that starts with the "#" symbol describes the data that follows. Usually, the chromosome (chr1 in this case) and position (10014903) are given, the general type of SV is reported (MEINFO=Alu), and more detailed data about the event is provided further. The Format column represents the names of statistic values that will be given for each sample for every loci, i.e. GT - genotype; GQ - genotype quality; SP - a number of correctly mapped read pairs that span breakpoint; FL - a call status (a flag) that tells if the call failed a particular filter; CN - number of subsets where the variant was observed. For example, for Alu.CHS sample, the statistics is 0/1:13:8:46:2.

```
chr13   59396895        59396896        SINE:Alu:AluYg6a2|279|-|IC:AC/2231:1105 66.88
chr5    114739736       114739737       SINE:Alu:AluUndef|258|-|IC:AC/2200:1114 66.39
chr1    100994221       100994222       SINE:Alu:AluUndef|273|+|IC:AC/2555:1351 65.41
chr4    67963892        67963893        SINE:Alu:AluUndef|266|-|IC:AC/2403:1375 63.61
chr11   48933125        48933126        SINE:Alu:AluYb6_2|254|-|IC:AC/1998:1210 62.28
```

Figure 3.3: **Excerpt of a BED file.** The chromosome name is given (e.g., chr13, chr5, etc.) in the first column. The approximate start and end positions of an event are reported in the second and third columns respectively (e.g., 59396895 - 59396896). This information is sufficient to run the tool. Since the example is taken from the dbRIP data, some information about the insertion is provided such as type (SINE:Alu), subfamily (AluYg6a2), length (279), strand (-), etc.

The core dataset of raw reads that are used to reconstruct the structural variant sequences is the 1000 Genomes Project phase 3 data [2]. It covers the genomes of more than 2,500 individuals from 26 distinct human populations representing all five major continents. It provides the largest amount of genome sequences, the best population diversity, and coverage of multiple sequencing platforms.

The input data files with raw reads are binary alignment (BAM) files that were sorted and merged from utilized data sets. The BAM file format is a binary format of the Sequence Alignment/Map (SAM) format which is a common high-throughput sequencing file format that is used for storing large nucleotide sequence alignments. It contains a header and an alignment section. Every header line starts with '@' sign and represents data about the version, the reference sequence, and the platforms that were used to produce reads. In the alignment section, every line represents an alignment of a fragment and has 11 mandatory fields. They include: a fragment (or query) name, a bitwise flag, a reference sequence name, a mapping position in a genome, a mapping quality, a CIGAR flag, a reference sequence of the mate, a mapping position of the mate, a mapping distance, a sequence, and sequencing base quality. There are some optional fields as well. The detailed specification and format description can be found in [41].

An example of SAM format is taken from the SAM specification [41] and is illustrated in Figure 3.4. Suppose we have the following alignment with bases in lower case clipped from the alignment. Read r001/1 and r001/2 is a read pair, r003 is a chimeric read, and r004 represents a split alignment.

```
Coor     12345678901234  5678901234567890123456789012345
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                      ATAGCT..............TCAGC
-r003                             ttagctTAGGC
-r001/2                                        CAGCGGCAT
```

Figure 3.4: **Example of alignment.** Image source: [41].

The corresponding SAM format is shown in Figure 3.5:

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

Figure 3.5: **Example of SAM format.** Image source: [41].

SAMtools [42] provide various utilities for alignment manipulation, including indexing by genomic position, fast retrieval of reads that align to a given location, sorting, and merging.

SAMtools were integrated into our program to efficiently collect required reads. There are 949 BAM files with data each of 70 GB size with the average coverage of around 30x. The duplications were removed and reads were sorted by genome location.

The list of MEIs is distributed over multiple threads and run simultaneously to increase the processing speed. For each MEI event, we are using SAMtools to collect discordant, concordant and split-reads within the specified flanking region - 600 bp before and after an insertion. We indicate the patterns of reads to search for by specifying the bitwise flags that are used by SAMtools for read description. Each bit in a flag provides different information about a read pair, such as the orientation of each mate, whether mates are properly aligned, whether a pair is concordant, etc. The SAM flags interpreter is available online at [28]. Table 3.1 represents the list of flags that were used for collecting the reads.

Table 3.1: SAMtools parameters.

Value	Description
-f 1	looking for paired reads
-F 4	excluding those with the first read unmapped
-F 8	excluding those with the second read unmapped
-F 256	excluding non-primary matches
-F 1024	excluding secondary alignment

Next, we apply the following filters to extract the discordant, concordant and

split-reads reads with good quality:

1. Each read pair is stored once.
2. The minimum read length (50 bp) should be met.
3. Reads with bad base quality are removed. The minimal quality for any individual base has to be greater than 10; the minimal average quality values across the entire read has to be greater than 20; the percentage of bases above the minimal quality for an individual base has to be at least 95; the minimal number of bases with quality above the minimal quality for an individual base has to be greater than or equal to 48.
4. For non-anchoring reads in a pair, the number of alternative alignments (represented by XA flag) has to be greater than or equal to two, and the number of suboptimal alignments (XS flag) has to be greater than or equal to five. This condition is checked for both reads in a pair.
5. If a read contains a lot of unknown bases (Ns) or the length of a read is too short (less than 50 bp), then they are determined to have bad sequencing quality and removed.
6. If a read pair is concordant and lies entirely either before or after the insertion, it is skipped.
7. Read pairs that flank the insertion site with no soft-clipping are skipped.

After filtering out reads with bad quality, we are checking for specific conditions to be met to extract those reads that will cover the breakpoint sequences and the insertion for further assembly. We are checking for the region a read belongs to, distance to the paired read, orientation of the read and the CIGAR flag.

These conditions can be divided according to the read pairs' type and relative position of the insertion. We keep read pairs that meet one of the following criteria:

1. Concordant pairs with the first (second) read split at the 5' side of the insertion point and the second (first) fully mapped to the 5' side (Figure 3.6). The 5' split-read has to meet the pattern (have a CIGAR flag) that will show first what number of bases are mapped to the reference and then what number of bases are soft clipped. The number of soft clipped bases has to be greater than 10. For example, the CIGAR flag is 70M30S which means that 70 bases of a read match and 30 bases are soft clipped.

2. Concordant pairs with the first (second) read split at the 3' side of the insertion point and the second (first) fully mapped to the 3' side (Figure 3.6). The 3' split-read has to meet the pattern that will show first what number of bases are soft clipped and then what number of bases are mapped to the reference. The number of soft clipped bases has to be greater than 10. For example, the valid CIGAR flag is 30S70M. In the case of soft clipping at the 3' end of the read, the mapping position is reported before the insertion position, even though the mapping part of a read starts after the insertion position.

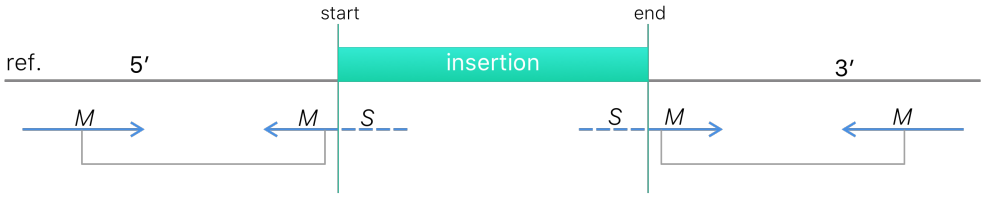

Figure 3.6: **Split-read 5' (3') pattern.** A gray line represents a reference genome; the green box is an insertion. Split-reads are represented by blue arrows. One mate is fully mapped to the reference genome, whereas the other is soft-clipped. Both reads are either in 5' or 3' end.

3. Concordant pairs with the first (second) read as the 5' split-read and the second (first) fully mapped to the 3' side, as shown in Figure 3.7.

4. Concordant pairs with the first (second) read as the 3' split-read and the second (first) mapped to the 5' side, as shown in Figure 3.7.

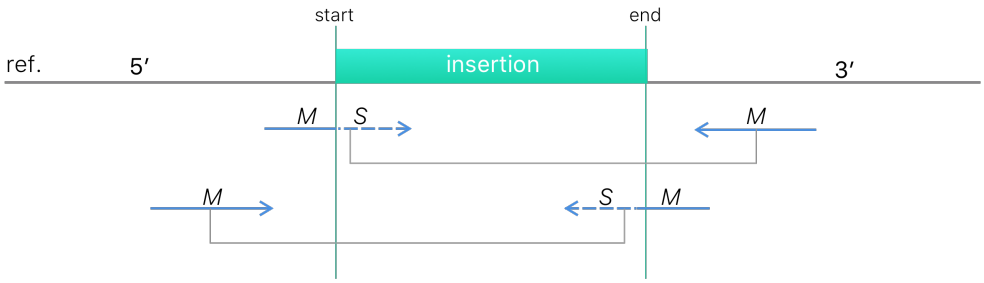

Figure 3.7: **Split in 5' (3'), mate in 3' (5') pattern.** A gray line represents a reference genome; the green box is an insertion. Split-reads are represented by blue arrows. One mate is fully mapped to the reference genome at 5' (3') end, whereas the other is soft-clipped at 3' (5') end.

5. Both are split-reads, as shown in Figure 3.8. The condition for a minimum number of soft clipped bases has to be met.

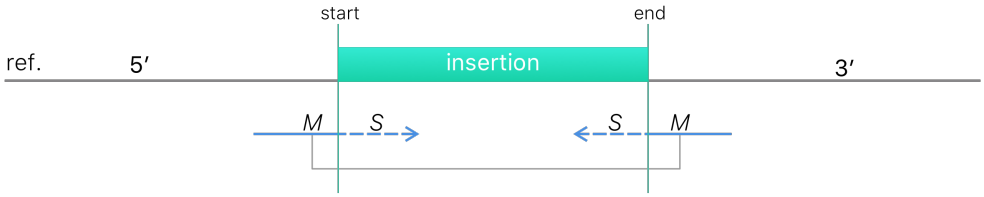

Figure 3.8: **Both mates are split-reads.** A gray line represents a reference genome; the green box is an insertion. A read pair is represented by blue arrows.

6. Pairs with the first (second) read as the 5' split-read and the mate mapped (fully or not) to a different chromosome or mapped at a distance greater than 1Mbp on the same chromosome. The mate is discordant in this case and considered to be inside the insertion, as shown in Figure 3.9.

7. Pairs with the first (second) read as the 3' split-read and mate inside the insertion (same as above), as shown in Figure 3.9.

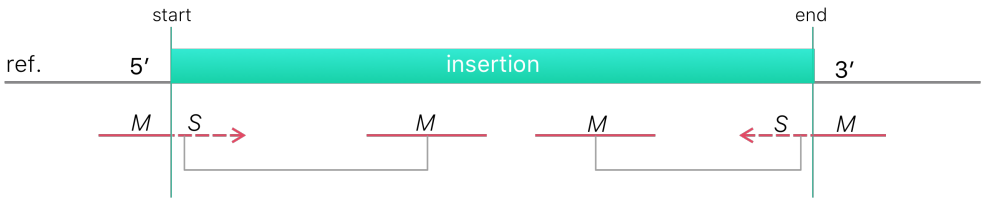

Figure 3.9: **Split in 5' (3'), mate discordant pattern.** A gray line represents a reference genome; the green box is an insertion. Discordant read pairs are represented by red arrows. One mate is soft-clipped from either side of the insertion, whereas the other is unaligned.

8. Discordant pairs with the first (second) read within the 5' flank and the mate inside the insertion, as shown in Figure 3.10.

9. Discordant pairs with the first (second) read in the 3' flank and the mate in the insertion, as shown in Figure 3.10.

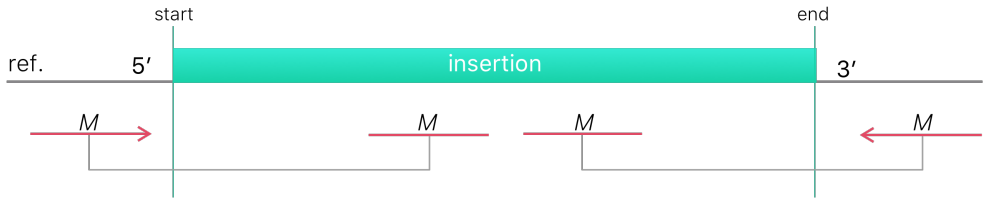

Figure 3.10: **Discordant reads pattern.** A gray line represents a reference genome; the green box is an insertion. Discordant read pairs are represented by red arrows. One mate is fully mapped to the reference genome and the other is unaligned.

3.3 Contigs assembly

The local assembly is performed using the CAP3 [25] tool to assemble the obtained valid reads into contigs that will cover the breakpoint sequences and ideally also the insertion. Before running the assembly, we remove redundant reads using the CD-HIT-EST program with the percentage of identity being equal to 0.98.

After testing different assemblers including Velvet [19], SOAPdenovo [45], SPAdes [10], and CAP3 [25], we found that CAP3 suits our needs better than the others, mainly because we can specify the overlap threshold and overlap percentage identity cutoff. In our case these values have to be low due to a high rate of sequencing errors. Table D.1 lists the parameters used to run the CD-HIT-EST and CAP3 tools. For a detailed description of these tools and parameters, refer to [25]. By default, the output will contain as many contigs as possible. The short contigs are hardly usable in our case, therefore, we introduced a threshold of 150 bp for each contig to be processed further.

Table 3.2: The CD-HIT and CAP3 parameters.

Tool	Parameter	Description
CD-HIT-EST	-c 0.98	percentage of identity
CAP3	-o 16	overlap length cutoff (in base pairs)
CAP3	-p 90	overlap percent identity cutoff
CAP3	-z 1	number of good reads at clip position (small because we removed redundant reads)

3.4 Bridge assembly

The main idea of our approach is to align and merge assembled contigs to the left and right flanking regions and extend the sequences from both sides as much as possible or until they overlap in the middle, like building a bridge (see Figure 3.11).

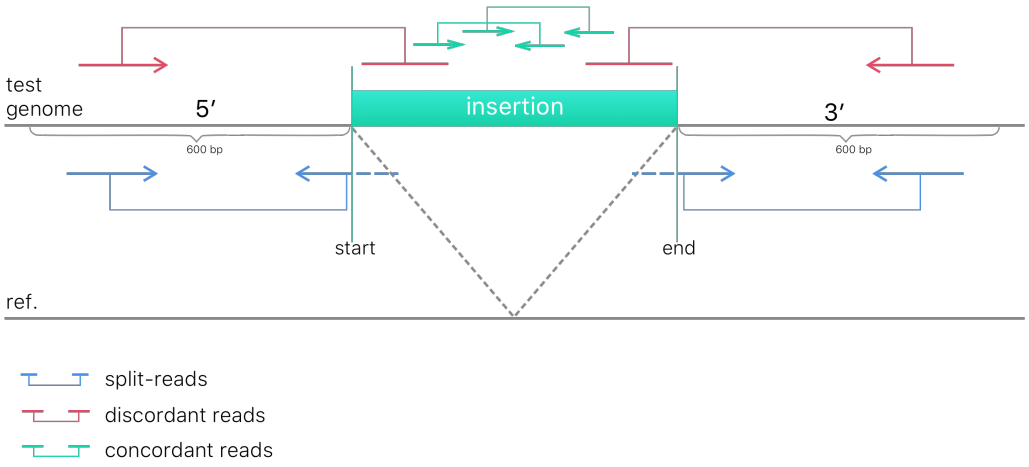

Figure 3.11: **"Bridge" assembly sketch.** The gray bottom line represents the reference genome; the gray upper line is a test genome. The green box represents the insertion. We work with the region of 600 bp before and after the insertion point. The blue arrows represent split-reads; the red lines (and arrows) are discordant reads; and the green arrows represent concordant reads. We collect and validate discordant and split-reads that lie in the region of interest. Then, we assemble them into contigs that ideally should cover the breakpoints' sequences. We extend the contigs from both sides of the insertion until they overlap. For long insertions, concordant reads that are mapped to the insertion have to be collected and added to the assembly process.

The implemented algorithm has several phases:

1. An undirected graph of overlapping contigs is built, and all possible paths between the left and the right flanking sequences are found.

2. The paths are validated, and the one that contains an insertion sequence is processed further.

3. If there is no complete path between two flanking regions, two trees are built for both sides, and the longest paths from the left and right flankings are searched.

4. Then, two opposite paths are aligned to the consensus database, and gaps are filled in.

5. In the end, the MEI information is collected, and TSD is deducted.

In the preprocessing step, all contigs are aligned to the reference flanking regions, and the ones that are fully mapped are filtered out as they contain only the flanking sequences.

During the first phase, all assembled contigs and two flanking sequences (600 bp from both sides of an insertion point) are aligned all-against-all in a pairwise fashion using the BLAST 2 SEQUENCES (*bl2seq*) program. *bl2seq* is a tool that uses the BLAST engine for pairwise sequence comparison [71].

These alignments are validated, and a non-symmetric adjacency matrix is built to represent the overlapping pairs. The value in the adjacency matrix for two sequences is the number of leftover bases from the alignment. Later, the path with the biggest weight, i.e. that describes the longest sequence, will be selected. The validation steps are illustrated in Figures 3.12-3.14 and are the following:

1. Aligning the left flanking sequence (query) to a contig (subject) case. The end of the alignment position in the query has to be at most 15 bases less than the end of the query, the leftover on the right side of the subject has to be greater than 10 bases, and the leftover on the left side of the subject has to be less than 50 bases. See Figure 3.12.

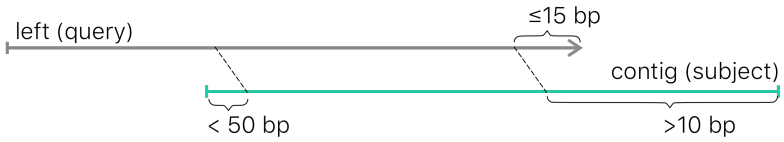

Figure 3.12: A valid alignment of the left flanking to a contig.

2. Aligning the right flanking sequence (query) to a contig (subject) case. The start position of the alignment in the query has to be maximum 15, the leftover on the left side of the subject has to be greater than 10 bases, and the leftover on the right side of the subject has to be less than 50 bases. As shown in Figure 3.13.

Figure 3.13: A valid alignment of the right flanking to a contig.

3. When two contigs are aligned to each other, the leftovers in both sides of the query's and subject's sequences cannot exceed 15 bases at the same time. Also, the length of the alignment has to be greater than 15 bases. See Figure 3.14.

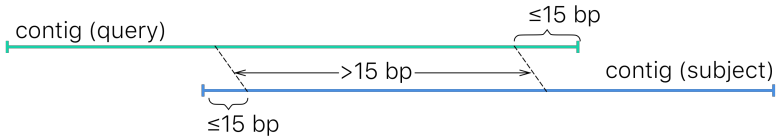

Figure 3.14: A valid alignment between two contigs.

An undirected graph is built, where nodes are contigs' and flanking sequences' identifiers, and edges connect two nodes if they have a valid overlap, i.e. the number of leftover bases are greater than 15. All possible paths from the left flanking to the right flanking sequences are found using a depth-first algorithm. After that, the paths are aligned to a type-specific consensus database - a database where all subtypes of one MEI type are annotated including the consensus sequences for each subtype. A consensus sequence (or canonical sequence) is a sequence that contains the most frequent nucleotides found at each position in multiple sequence alignments. The Basic Local Alignment Search Tool (BLAST) [7] (the command line version of the *blastn* program for nucleotide alignment) is integrated to map contigs to the consensus databases of MEIs. Before the alignment, the MEI consensus database has to be indexed. The parameters used to run BLAST are listed in Table 3.3.

The alignment results include the sequence description (subtype name in our case) to which a query sequence was aligned, start and end positions of the alignment in the given sequence and in the consensus sequence, a percentage of identical matches, an alignment length, a number of mismatches and a number of gaps, as well as an expected value and a bit score. The alignment is considered to be valid when the percentage of matches is greater than 90%, and the length of the alignment is above the minimum insertion length specified for each type of MEI. The path with the

Table 3.3: BLAST parameters.

Parameter	Value	Description
-db	.fa file	link to a consensus database
-query	.fa file	a FASTA file with a sequence
-max_target_seqs	1	number of aligned sequences to keep
-perc_identity	90	percent identity cutoff
-outfmt	6	alignment view options
-evalue	1e-5	expect value for saving hits
-word_size	9	length of initial exact match
-F	F	for not filtering out the low complexity sequence match

best alignment to the consensus database is selected, and the MEI information is extracted from the alignment data. The information we can obtain from the BLAST output data is the subtype of the MEI, the length of the identified insertion, and the orientation of the insertion sequence.

If there is no such a path that will connect the left flanking sequence with the right, then we build two trees with the roots at the left flanking and the right flanking. Then, we search for the longest paths with the largest weight between the roots and any leaf in the trees using a pre-order depth-first traversal algorithm. Figure 3.15 is an example for the left flanking tree creation. After that, these two paths are aligned to the consensus database. If they are both aligned, the information about the MEI is obtained and the insertion sequence is filled with Ns where there is a gap between the left and the right paths with the length of the gap calculated as the distance between the two aligned regions on the consensus sequence. If the alignments of the paths overlap, we can reconstruct the full insertion sequence using the alignment positions.

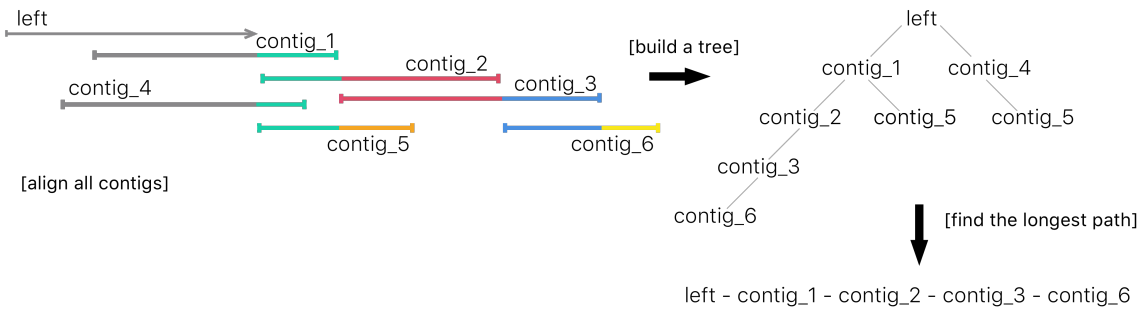

Figure 3.15: **Example of the left tree.** A gray top left arrow represents the left flanking sequence. The lines below are contigs. The assembled contigs are aligned in a pairwise fashion, and their overlapping scores are stored. A tree with a root in the left flanking is built, and the path with the biggest total score from the root to a leaf is found.

3.5 MEI information collection

The last step involved in the characterization process is a collection of the MEI information including the type, the orientation, and the length. Also, associated local rearrangements such as the target site duplications (TSD), insertion-mediated deletions (IMD), and 5' and 3' transductions are deduced where possible. Insertion-mediated deletions are the target site deletions that happened upon insertion. Transductions are extra sequences that carried from the parent sequence during insertion and are attached to the 5' or 3' flanking region. Figure 3.16 illustrates the variants of local rearrangements.

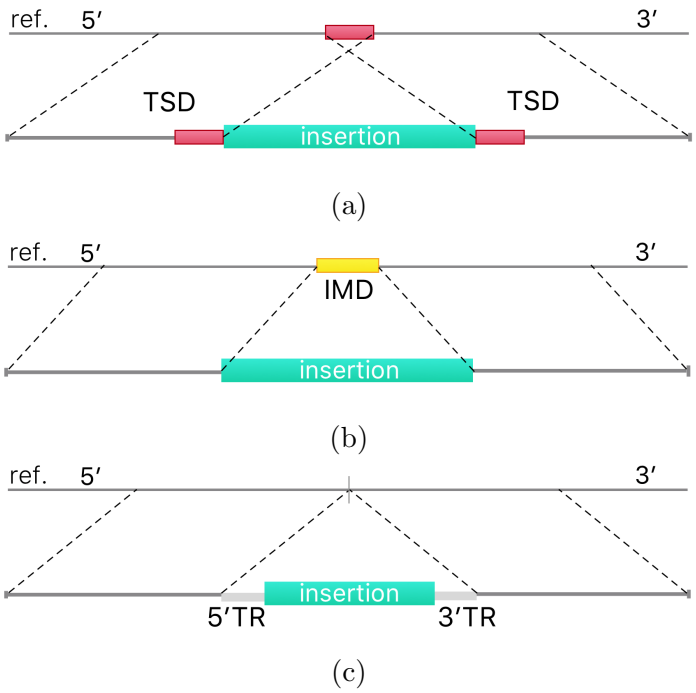

Figure 3.16: **(a) TSD, (b) IMD, and (c) transductions.** The local rearrangements that occur at the ends of MEIs can be detected by aligning the obtained sequence to the reference genome. In case of TSD, the alignment locations will overlap. When there is a gap between the alignments of two flanking regions, IMD occurred. When the alignments of two flanking sequences are adjacent but there are extra bases between the insertion and the flank(s), we can deduce 5' and/or 3' transductions.

After aligning the obtained sequence (or sequences) to the ME consensus sequence database, we process the BLAST output file to analyze the alignment data and collect necessary information about the MEI. An example of the BLAST output is shown in Figure 3.17. We collect entries that contain matches with at least 96% sequence similarity to the query sequence. If there are multiple valid alignments to different consensus sequences, we store them all and the one with the highest percentage of similarity is processed further. From the mapped positions in the consensus we can extract the orientation (e.g., if the end position is less than the start then the orientation is reversed), the length of the insertion, and the type (i.e. the consensus identifier to which the insertion was mapped).

query id	subject id	% identity	length of alignment	# of mis-matches	# of gaps	query start	query end	subject start	subject end	expect value	bit score
left_flank_Contig3_right_flank	AluYa5	99.29	283	1	1	641	923	282	1	2e-147	510

Figure 3.17: **Example of blastn alignment output.** The alignment information contains the query identifier (the name of our sequence), the subject identifier (in this case the subtype name, AluYa5), the length of the alignment (283), the number of mismatches and gaps, and the start and end alignment positions in the query (641 - 923) and in the subject (282 - 1).

To find the TSD sequence, we run the BLAST tool with the full sequence that contains the two flanking regions and the insertion against the reference sequence of the corresponding chromosome. We keep the alignments that are close to the insertion point in the reference and sort their mapping positions. If two alignments overlap so that the start position of the second one is less than the end position of the first one, then the TSD is present and the boundaries are the described positions. If the two alignments do not overlap, the end of the first one and the start of the second one frame an insertion-mediated deletion.

3.6 Output format

There are two different types of output: (i) for characterized MEIs, and (ii) for MEIs that cannot be characterized for some reason. Both are FASTA files.

For characterized MEIs, the description line format is represented in Figure 3.18. The left flanking sequence is included after the description. If TSD is found, it is also included, otherwise empty line is added. Insertion sequence is added after that. TSD is repeated again. Finally, the right flanking is included at the end of the sequence. After the insertion allele, the pre-integration allele is reported. The pre-integration allele includes the left flanking region from the reference, TSD, and the right flanking. In the end, "//" symbols are included to separate events.

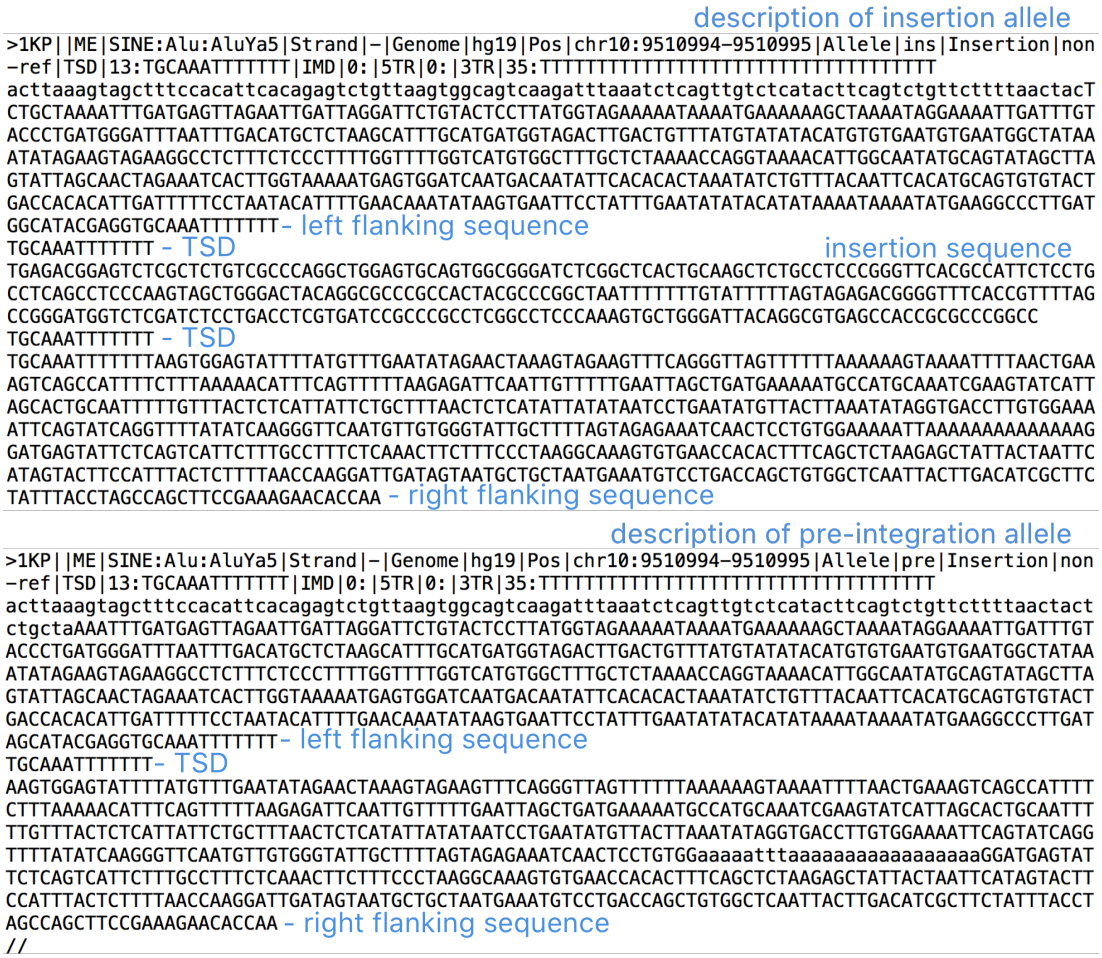

Figure 3.18: **Example of the successfully characterized MEI output.** For the insertion allele, firstly, the description of the MEI is provided. It contains chromosome, position, type, strand, the reference genome that was used, TSD, IMD, and transductions. Secondly, the left flanking sequence is reported. After that, TSD is included if it was deduced. Then, the reconstructed insertion sequence is reported. Another copy of TSD and the right flanking sequence are written at the end. For the pre-integration allele, the information remains the same, except the insertion is not included.

For non characterized MEIs, the description format includes chromosome and position in the reference genome, the data about the alignment to the consensus database (if available), and the full path of contigs that created the extended left and right regions. After the description, the sequence of the merged path is reported. For each event, both paths from the left and right are included and each event is separated by "//" symbols. An example of non characterized MEI format is shown in Figure 3.19.

```
>1KP|ME|SINE:Alu:AluYa4|1-135|Strand|+|Genome|hg19|chr1:7242044-7242044|Allele|ins|left_flank_Co
ntig3  - description of left sequence
tagcaattaggtaagataaatcagttactggcatgaatattagaaaagaagaaccagaactatctcagtttgcacatgatatgatagtatactggg
ataaccctagagaatcaatgataaaagtaaattaaacaataaaatcattatggtagcaggaaaaatttaacatatagaacttattaactttcttgt
acataaataagaaccacataaataaccagtttgaggacgcaatagcaacaaagaagatacttaggaataaacttagaaagatatgggcaaagcctt
gataagaaacatttttaagctcttctgaaagatgtgaaagtaAGTCTTGAACAAATAGAAAGACATCCCTTGATTTTGGATGGATTGATTTAACAT
CATAAAAAAGTTTGTCCCAAGTTAATTTATGAATTTAACATAATCCCAGAGAAAGTATCAGTAAGTTTTTTTATGGAGCTACACAACTGATACTTA
AGTTCATATGAAAAGACAAACATGCAGTAACCACGATAACCCTGCAAATTAAGAACTACTGGGAGGAATAGCTCTACCAAACATTAAAATGAACTA
TAAAGCCTCTATAATTAAAACAGTAGGCCGGGCGCGGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGCCGAGGCGGGCGGATCACGAGGTCA
GGAGATCGAGACCATCCCGGCTAAAACGGTGAAACCCCGTCTCTACTAAGAATATAAAAAATTA - left merged path
>1KP|ME|SINE:Alu:Undef|Strand|Undef|Genome|hg19|chr1:7242044-7242044|Allele|ins|right_flank_Cont
ig5 - description of right sequence
AGTGTAGGTTCTCACTACTGGGAGGAATAGCTCTACCAAACATTAAAATGAACTATAAAGCCTCTATAATTAAAACAGTATGGTGAAATTGATGCA
TGAATATTCAAAATAGCAGTAGAATTGAATAGAAAGTCTTGAAATAGACCCCAAGTTCATATGGAAATTTAGTATATGACAAAGGTGGTATCTCAA
ATCACTGGAGCAGAGATTACCTTTTCAATACATTGTTCCGGGATGACCAGGCAGCCTTTGGAAGAAGATAAAATTAATGCATAAGTTCCAGTATAC
ACAAGAAAAACTCCAAATGGATTGAGATCTAAATATAAAAGGTGAAATCATAAAAGTACTAGAAGTGGCTGGGCACAGTGGCTCATGCCTGTAATC
CTTGCACTTTGGGGGGCCAAGGCAGGTGGATTGCTTGAGCCCAGGAGTTCGAGACTAGCCTGGGCAACATGGCAAAACCCTGTCTCTACTAAAAAT
ACAAAAACTAAGGTGGGAGGATCACCTGAGCCTGGGGAGGTCGAGGCTGCAGTGAACCGTGATCGTGCCACTGTACTCCGGCCTGGGTGACAGAGT
GAACACCCCATCTCAGAAAAAAAAAAAAACAAAACACTAGAAGAAAACATGGGTGAGCTTTTCTGTAACAggatgtaagaaaaggttttctaacaat
gactcaa - right merged path
//
```

Figure 3.19: **Example of failed characterization output.** For the failed MEI, we store the full information we could collect, such as type and strand. We report the paths with contigs names from both sides of the insertion and the corresponding merged contigs.

3.7 Validation

Validation is done manually in this work by randomly picking characterized MEIs and assessing the patterns provided by the online version of the BLAT and BLASTN tools.

The BLAT tool is accessible via the USCS Genome browser [35] under Tools - Blat section. The alignment data has information about the chromosome, the strand to which the sequence was mapped, the start and end positions in the sequence and in the chromosome, the alignment score, and the length. An example of the alignments produced by BLAT is shown in Figure 3.20.

ACTIONS	QUERY	SCORE	START	END	QSIZE	IDENTITY	CHRO	STRAND	START	END	SPAN
browser details	MEI	1192	1	1521	1521	98.5%	10	+	9468431	9469630	1200
browser details	MEI	328	564	927	1521	98.6%	10	-	49069955	49070482	528
browser details	MEI	325	589	923	1521	98.9%	3	-	177767233	177767572	340

Figure 3.20: Example of the list of alignments generated by BLAT.

For the sequences that contain the insertion, there will be two alignments reported at the top. The first one is the alignment of the flanking regions (see Figure 3.21) and the second one is the insertion (see Figure 3.22). The blue letters in Figures 3.21 and 3.22 represent aligned bases and the black ones represent bases that are not

aligned. The insertion part will have multiple alignments to different chromosomes of approximately the same length.

The browser version of the BLASTN program [58, 83] gives a visual representation of the sequence's alignment. The properly reconstructed MEI will have uniquely (in most cases) mapped flanking regions and multiple alignments for the insertion, as shown in Figure 3.23.

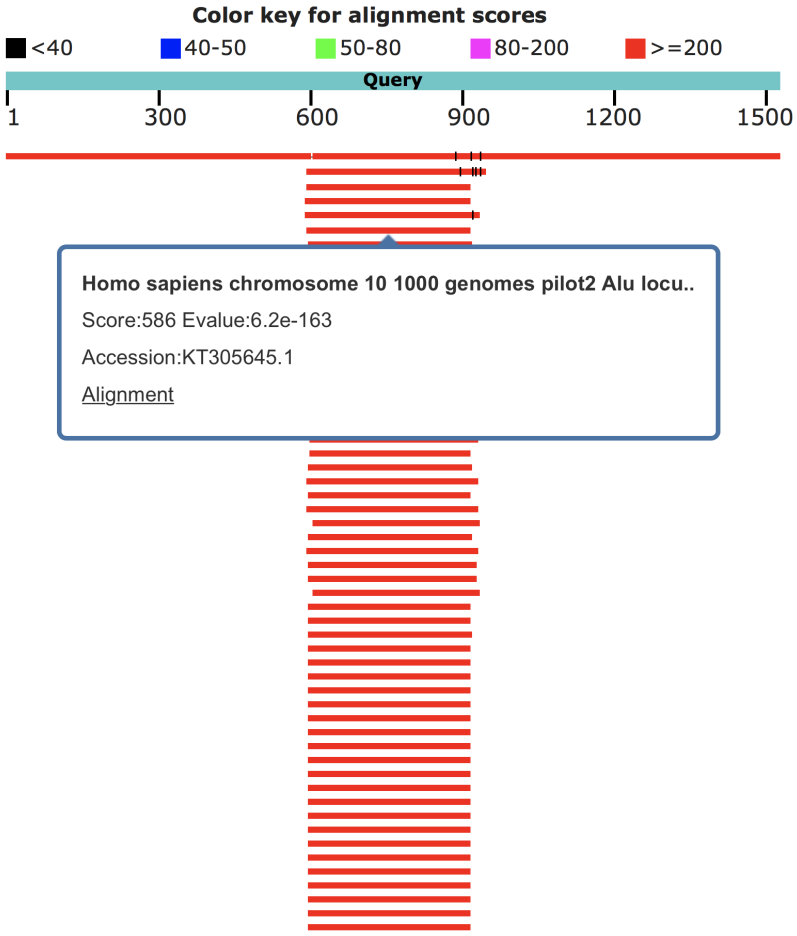

Figure 3.23: **BLASTN alignment of the full sequence.** The top red line represents the full sequence that contains two flanking regions and the insertion. The flanking sequences are uniquely aligned with the alignment score $\geqslant 200$. The insertion part has multiple alignments to different locations in the genome with high alignment scores. It indicates the repetitive nature of MEI. Therefore, the reconstructed sequence can be considered as valid.

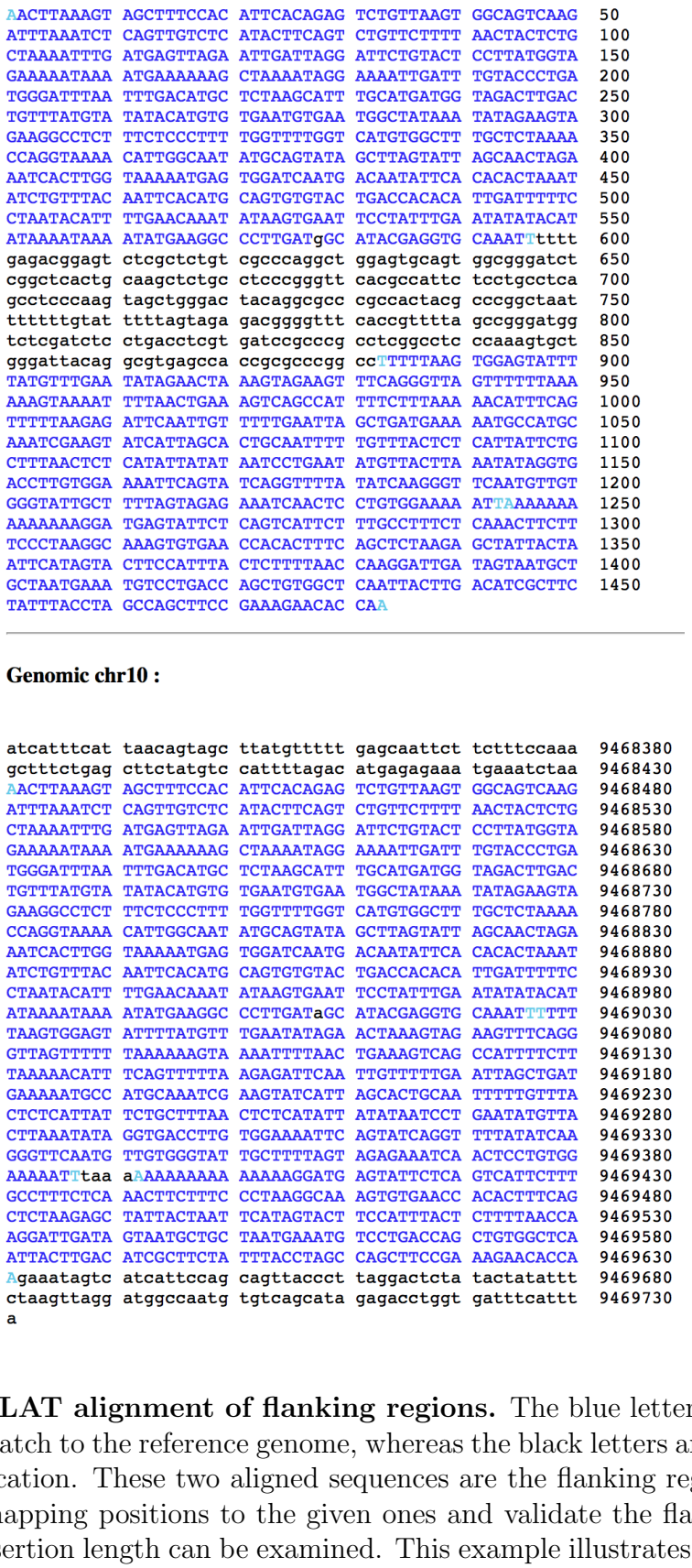

Figure 3.21: **BLAT alignment of flanking regions.** The blue letters in the alignment output match to the reference genome, whereas the black letters are not mapped to the same location. These two aligned sequences are the flanking regions. We can compare the mapping positions to the given ones and validate the flanking regions. As well, the insertion length can be examined. This example illustrates Alu insertion. Since we know the average size of Alu elements, we can conclude that our insertion sequence is fully characterized.

```
aacttaaagt agctttccac attcacagag tctgttaagt ggcagtcaag  50
atttaaatct cagttgtctc atacttcagt ctgttctttt aactactctg  100
ctaaaatttg atgagttaga attgattagg attctgtact ccttatggta  150
gaaaaataaa atgaaaaaag ctaaaatagg aaaattgatt tgtaccctga  200
tgggatttaa tttgacatgc tctaagcatt tgcatgatgg tagacttgac  250
tgtttatgta tatacatgtg tgaatgtgaa tggctataaa tatagaagta  300
gaaggcctct ttctcccttt tggttttggt catgtggctt tgctctaaaa  350
ccaggtaaaa cattggcaat atgcagtata gcttagtatt agcaactaga  400
aatcacttgg taaaaatgag tggatcaatg acaatattca cacactaaat  450
atctgtttac aattcacatg cagtgtgtac tgaccacaca ttgatttttc  500
ctaatacatt ttgaacaaat ataagtgaat tccTATTTga aTATATACAT  550
ATAaaataaa atatgaaggc ccttgatggc atacgaggtg caaaTTTTTT  600
GAGACGGAGT CTCGCTCTGT CGCCCAGGCT GGAGTGCAGT GGCGGGATCT  650
CGGCTCACTG CAAGCTCtGC CTCCCGGGTT CACGCCATTC TCCTGCCTCA  700
GCCTCCCAAG TAGCTGGGAC TACAGGCGCC CGCCACTACG CCCGGCTAAT  750
TTTTTtGTAT TTTTAGTAGA GACGGGGTTT CACCGTTTTA GCCGGGATGG  800
TCTCGATCTC CTGACCTCGT GATCCGCCCG CCTCGGCCTC CCAAAGTGCT  850
GGGATTACAG GCGTGAGCCA CCGCGCCCGG CCTTTTTaag tggagtattt  900
tatgtttgaa tatagaacta aagtagaagt ttcagggtta gtttttaaa  950
aaagtaaaat tttaactgaa agtcagccat tttctttaaa aacatttcag  1000
tttttaagag attcaattgt ttttgaatta gctgatgaaa aatgccatgc  1050
aaatcgaagt atcattagca ctgcaatttt tgtttactct cattattctg  1100
ctttaactct catattatat aatcctgaat atgttactta aatataggtg  1150
accttgtgga aaattcagta tcaggtttta tatcaagggt tcaatgttgt  1200
gggtattgct tttagtagag aaatcaactc ctgtggaaaa attaaaaaaa  1250
aaaaaaagga tgagtattct cagtcattct ttgcctttct caaacttctt  1300
tccctaaggc aaagtgtgaa ccacactttc agctctaaga gctattacta  1350
attcatagta cttccattta ctcttttaac caaggattga tagtaatgct  1400
gctaatgaaa tgtcctgacc agctgtggct caattacttg acatcgcttc  1450
tatttaccta gccagcttcc gaaagaacac caa
```

Genomic chr11 :

```
gactgtattt ttaatgaaac atctcatgaa atcaggattt catgggtttc  41122846
agaaaatgat gcttttgtta tactaaacaa tttattttat aattttattg  41122896
TATTTTATAT ACATATAtac tgacacttaa aaaatttgtg tgtcatcctt  41122946
gtatggtggt catgctagtc ttctccgtat tgttccaact ttagtacgta  41122996
ttcagcctaa gtgagcacta tactaaatat ttcctagcag tcttgagtat  41123046
gtatgttgaa ttgttggaag aacttgagat catccaataa aaactccaac  41123096
acaactggaa attcccatcc tatgcaagtg tggtgagaga ggtcaggcag  41123146
cctctcttga agcacctcca aggagggtaa attccctggt cagtgagtca  41123196
tcatatcttg atttcaagca actctggatt ttataaatgc tttctctatc  41123246
ctgagctatg ttttgttttt ttttttttt tttttttTT TTTTGAGACG  41123296
GAGTCTCGCT CTGTCGCCCA GGCTGGAGTG CAGTGGCGGG ATCTCGGCTC  41123346
ACTGCAAGCT CcGCCTCCCG GGTTCACGCC ATTCTCCTGC CTCAGCCTCC  41123396
CAAGTAGCTG GGACTACAGG CGCCCGCCAC TACGCCCGGC TAATTTTTTG  41123446
TATTTTTAGT AGAGACGGGG TTTCACCGTT TTAGCCGGGA TGGTCTCGAT  41123496
CTCCTGACCT CGTGATCCGC CCGCCTCGGC CTCCCAAAGT GCTGGGATTA  41123546
CAGGCGTGAG CCACCGCGCC CGGCCTTTTT tttcttttt taataaattc  41123596
taccactggt tctagctttg tcctcagaag aacacaggat aaattaattc  41123646
attctttccc agcttctctg agttcactca
```

Figure 3.22: **BLAT alignment of the insertion part.** The blue letters in the alignment output match to the reference genome, whereas the black letters are not mapped to the same location. The alignment of the insertion part is represented here. We can observe the poly T tail before the insertion, which is a sign for MEI. As well, there are multiple alignments with approximately the same scores. Therefore, we can conclude that the sequence is an MEI.

Chapter 4

Results

We tested our program on the list of non-reference MEI events annotated in the Retrotransposon Insertion Polymorphism in Humans (dbRIP) database. The dbRIP database contains the full sequences of the non-reference alleles for a total of 1,087 locations covering 862 Alus, 18 SVAs, 203 L1s, and 4 LTRs. For dbRIP data, our system achieved a sensitivity value of 54% for Alu, 72% for SVA, and 72% for L1.

We worked with a non-redundant list of 1,613 non-reference MEIs locations from the 1kGP data. We were able to characterize 862 out of 1,372 (62.76%) Alu entries, 30 out of 95 (31.58%) SVAs, and 39 out of 146 (26.71%) L1s.

The step by step description of intermediate results is provided below and the pitfalls are discussed.

4.1 Successful MEI characterization

We will demonstrate the results of a successful characterization by analyzing the intermediate output at each step of the process. As an example, a random successfully characterized MEI is picked.

For Alu characterization, we will discuss the outputs for the event at chromosome 10, position bp 9,510,994. The flanking region is 600 bp before and 600 bp after the insertion point. This means we are working with an interval of 1,200 bp from 9,510,394 to 9,511,594. The left and the right flanking sequences are extracted from the reference genome from the corresponding region and are equal in length.

To collect reads from utilized data from different genomes, we run SAMtools as was described earlier. Overall, there are 297,737 raw reads that cover the region of interest. However, only 1,779 reads are valid and are considered to support the insertion and the breakpoints. The coverage is equal to 145x, which is enough to

consider the set of reads to be sufficient for the assembly. The reads are written down into a separate FASTA format file in paired fashion, i.e. the first read is followed by the second one in the same file and suffixes "_1" and "_2" are added to the reads' identifiers respectively, corresponding to a specific MEI event. An example of a valid pair is shown in Figure 4.1.

```
>ERR1044266.70949297_1 chr10:9510982|+|CHB70G_p28
CCTCCCAAAGTGCTGGGATTACAGGCGTGAGCCACCGCGCCCGGCCTGCAAATTTTTTTAAGTGGAGTATTTTATGTTTGAATATAGAACTAAAGTAGAA
>ERR1044266.70949297_2 chr10:9511001|-
GTTTTTAAAGAAAATGGCTGACTTTCAGTTAAAATTTTACTTTTTTAAAAAACTAACCCTGAAACTTCTACTTTAGTTCTATATTCAAACATAAAATACT
>ERR1044405.39582164_1 chr10:9510910|+|CHB70G_p88
AAATATAAGTGAATTCCTATTTGAATATATACATATAAAATAAAATATGAAGGCCCTTGATGGCATACGAGGTGCAAATTTTTTTTTTTTTTTTTTTTT
>ERR1044405.39582164_2 chr5:88043165|-
GAACCCGGGAGGCAGAGCTTGCAGTGAGCCGAGATCCCGCCACTGCACTCCAGCCTGGGCGACAGAGCGAGACTCCGTCTCAAAAAAAAAAAAAAAAAA
>FCD19JBACXX:6:2310:2579:36380#_1 chr10:9511240|-|ASD_C1.1989-24099
AGGAGTTGATTTCTCTACTAAAAGCAATACCCACAACATTGAACCCTTGATATAAAACCTGATACTGAATTTTCCACAAGGTCACCTATATTTAAGTAAC
>FCD19JBACXX:6:2310:2579:36380#_2 chr6_cox_hap2:826354|-
GAGACGGGGTTTCACCGTTTTAGCCGGGATGGTCTCGATCTCCTGACCTCGTGATCCGCCCGCCTCGGCCTCCCAAAGTGCTGGGATTACAGGCGTGAGC
>ERR1055505.17315973_1 chr10:9511247|-|CHB70G_p18
TTTCCACAGGAGTTGATTTCTCTACTAAAAGCAATACCCACAACATTGAACCCTTGATATAAAACCTGATACTGAATTTTCCACAAGGTCACCTATATTT
>ERR1055505.17315973_2 chr2:188649809|+
GGCTAATTTTTTTGTATTTTTAGTAGAGACGGGGTTTCACCGTTTTAGCCGGGATGGTCTCGATCTCCTGACCTCGTGATCCGCCCGCCTCGGCCTCCCA
```

Figure 4.1: **Example of qualified reads in a FASTA file.** In the description line, first, the read pair's identifier is reported with "_1" or "_2" specification for the first and the second read in a pair respectively. Then, the loci and the strand of the insertion is written, and the data source file name is stored. After the description, is the read's sequence.

The collection and validation of reads are the most time-consuming parts of the pipeline. For the chosen loci, it took two and a half hours to process all 949 BAM files with data and extract the necessary reads. On average, it takes two to three hours for every location. The collected reads are stored based on chromosome and position information. Therefore, the assembly part can be run and tested independently in a reasonable time.

To speed up the collection of reads we introduced the split-reads cutoff. During the validation phase we calculate the number of split-reads that cover the breakpoints and the number of reads that are inside the insertion, and when we reach a sufficient threshold and coverage, we stop processing the data files. This approach can reduce the running time by half.

After the reads are collected, we run the CD-HIT-EST clustering tool to remove redundant reads with the similarity threshold set to 0.98. The input is a FASTA file created previously and the output are two files: a FASTA file of representative sequences and a text file of a list of clusters. The threshold was selected after a bunch of test runs. Since the level of diversity is high and we are dealing with repetitive sequences, the threshold should be low enough to catch the alignment in

the problematic regions.

Next, we assemble selected reads using the Contig Assembly Program (CAP3) which is designed for small-scale assembly. As an input it takes the FASTA file of sequences produced by CD-HIT. The output consists of several files: (i) *.contigs* file where consensus sequences are saved, (ii) *.contigs.qual* that stores quality values of consensus sequences, (iii) *.capout* where the detailed alignments of assembled sequences in each contig are reported, (iv) *.singlets* that stores reads that were not used in assembly, and (v) *.info* file with additional information about assembly. An example of the general output is illustrated in Figure 4.2.

In the initial assembly, overlaps of minimum specified length (in our case 16) and a minimal overlap percentage identity cutoff (in our case 90) are found. After that, two contigs are merged if the overlapping alignment score is greater than or equal to a specified threshold.

The assembled contigs that are longer than 150 bp are kept and stored separately in individual files.

```
>Contig1
TTAATTAGCACGACTCATATTAGAGTCTTGGATAAGGGCCTAAGAGTCTTTTAAGTGGCA
GTCAAGATTTAAATCTCAGTTGTCTCATACTTATTATATCCCAAAAATACATAGATCGGA
AG
>Contig2
GAATACTCATCCTTTTTTTTTTTTTTAATTTTTCCACAGGAGTTGATTTCTCTACTAAAA
GCAATACCCACAACATTGAACCCTTGATATAAAACCTGATACTGAATTTTCCACAAGGTC
ACCTATATTTAAGTAACATATTCAGGATTATATAATATGAGAGTTAAAGCAGAATAATGA
GAGTAAACAAAAATTGCAGTGCTAATGATACTTCGATTTGCATGGCATTTTTCATCAGCT
AATTCAAAAACAATTGAATCTCTTAAAAACTGAAATGTTTTTAAAGAAAATGGCTGACTT
TCAGTTAAAATTTTACTTTTTTAAAAAACTAACCCTGAAACTTCTACTTTAGTTCTATAT
TCAAACATAAAATACTCCACTTAAAAAAATTTGCAGGCCGGGCGCGGTGGCTCACGCCTG
TAATCCCAGCACTTTGGGAGGCCGAGGCGGGCGGATCACGAGGTCAGGAGATCGAGACCA
TCCCGGCTAAAACGGTGAAACCCCGTCTCTACTAAAAATACAAAAAAATTAGCCGGGCGT
AGTGGCGGGCGCCTGTAGTCCCAGCTACTTGGGAGGCTGAGGCAGGAGAATGGCGTGAAC
CCGGGAGGCAGAGCTTGCAGTGAGCCGAGATCCCGCCACTGCACTCCAGCCTGGGCGACA
GAGCGAGACTCCGTCTCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAA
>Contig3
GCTGGAATGATGACTATTTCTTGGTGTTCTTTCGGAAGCTGGCTAGGTAAATAGAAGCGA
TGTCAAGTAATTGAGCCACAGCTGGTCAGGACATTTCATTAGCAGCATTACTATCAATCC
TTGGTTAAAAGAGTAAATGGAAGTACTATGAATTAGTAATAGCTCTTAGAGCTGAAAGTG
TGGTTCACACTTTGCCTTAGGGAAAGAAGTTTGAGAAAGGCAAAGAATGACTGAGAATAC
TCATCCTTTTTTTTT
>Contig4  - not long enough
GCAGCAATACTTTATTCTCAAAGTGAAGTTTTAGGCAGTCAAGATTTAAATCTCAGTTGT
CTCATACTTCAGTCTGTTCTTTTAACTACTCT
>Contig5
TGTATACATTAGAAAAGCAGTTTGGTGTCTCCTATGACTCTCTGCTAAAATTTGATGAGT
TAGAATTGATTAGGATTCTGTACTCCTTATGGTAGAAAAATAAAATGAAAAAAGCTAAAA
TAGGAAAATTGATTTGTACCCTGATGGGATTTAATTTGACATGCTCTAAGCATTTGCATG
ATGGTAGACTTGACTGTTTATGTATATACATGTGTGAATGTGAATGGCTATAAATATAGA
AGTAGAAGGCCTCTTTCTCCCTTTTGGTTTTGGTCATGTGGCTTTGCTCTAAAACCAGGT
AAAACATTGGCAATATGCAGTATAGCTTAGTATTAGCAACTAGAAATCACTTGGTAAAAA
TGAGTGGATCAATGACAATATTCACACACTAAATATCTGTTTACAATTCACATGCAGTGT
GTACTGACCACACATTGATTTTTCCTAATACATTTTGAACAAATATAAGTGAATTCCTAT
TTGAATATATACATATAAAATAAAATATGAAGGCCCTTGATGGCATACGAGGTGCAAATT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

Figure 4.2: **The CAP3 output "Alu.chr10_9510994.cdhit.cap.contigs" file.** The output file contains the description (or name) given to the merged contig. The description line starts with the ">" sign. The actual sequence follows the description.

The next step is to filter out contigs that fully match the flanking region(s). We run the BLAST program to align each contig to the flanking reference sequence. The output file is in a FASTA format and contains the names of sequences, start and end positions of the alignment, as well as the percentage of identity and gaps found. The reads that are aligned to the reference entirely and with the percentage of alignment greater than 90% are removed from the list.

There are six assembled contigs for the chosen loci, and four of them meet the requirements.

We add two flanking sequences to the set of contigs before processing further. Having the qualified set of assembled contigs and flanking regions that ideally will cover the breakpoints sequences and insertion, we align them against each other in a

pairwise fashion using the *bl2seq* program. The conditions of valid alignments were described in the previous chapter. An example of *bl2seq* output of the valid alignment is provided in Figure 4.3. It has contig names, lengths, and the alignment information. There can be several alignments, thus we are looking for the best one and assess it. The alignment output shows the percentage of identity, the length in base pairs, the orientation of each contig, and the start and end positions in each sequence. Then the alignment is displayed at nucleotide level.

```
Query= left_flank
         (600 letters)



>Contig5
          Length = 574

 Score =  965 bits (502), Expect = 0.0
 Identities = 504/505 (99%)
 Strand = Plus / Plus


Query: 96  ctctgctaaaatttgatgagttagaattgattaggattctgtactccttatggtagaaaa 155
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 40  ctctgctaaaatttgatgagttagaattgattaggattctgtactccttatggtagaaaa 99


Query: 156 ataaaatgaaaaaagctaaaataggaaaattgatttgtaccctgatgggatttaatttga 215
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 100 ataaaatgaaaaaagctaaaataggaaaattgatttgtaccctgatgggatttaatttga 159


Query: 216 catgctctaagcatttgcatgatggtagacttgactgtttatgtatatacatgtgtgaat 275
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 160 catgctctaagcatttgcatgatggtagacttgactgtttatgtatatacatgtgtgaat 219


Query: 276 gtgaatggctataaatatagaagtagaaggcctctttctccctttggttttggtcatgt 335
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 220 gtgaatggctataaatatagaagtagaaggcctctttctccctttggttttggtcatgt 279


Query: 336 ggctttgctctaaaaccaggtaaaacattggcaatatgcagtatagcttagtattagcaa 395
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 280 ggctttgctctaaaaccaggtaaaacattggcaatatgcagtatagcttagtattagcaa 339


Query: 396 ctagaaatcacttggtaaaaatgagtggatcaatgacaatattcacacactaaatatctg 455
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 340 ctagaaatcacttggtaaaaatgagtggatcaatgacaatattcacacactaaatatctg 399


Query: 456 tttacaattcacatgcagtgtgtactgaccacacattgatttttcctaatacattttgaa 515
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 400 tttacaattcacatgcagtgtgtactgaccacacattgatttttcctaatacattttgaa 459


Query: 516 caaatataagtgaattcctatttgaatatatacatataaaataaaatatgaaggcccttg 575
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 460 caaatataagtgaattcctatttgaatatatacatataaaataaaatatgaaggcccttg 519


Query: 576 atagcatacgaggtgcaaatttttt 600
           || ||||||||||||||||||||||
Sbjct: 520 atggcatacgaggtgcaaatttttt 544
```

Figure 4.3: **The *bl2seq* output example for the left flanking sequence and contig_5.** The alignment information consists of names of query and subject, the percentage of identities, the orientation of the alignment, the positions, and the sequence alignment at the nucleotide level.

The information about every properly overlapping pair is stored in a non-symmetric adjacency matrix, where a non-zero value in row i column j represents a leftover in sequence j from aligning sequence i to sequence j, and "0" if the alignment is not valid or there is no leftover:

Table 4.1: The adjacency matrix for the Alu chr10_9510994.

	left	**contig_4**	**contig_2**	**contig_3**	**contig_6**	**right**
left	0	0	0	0	36	0
contig_4	0	0	0	698	0	0
contig_2	0	0	0	0	0	0
contig_3	0	0	0	0	538	341
contig_6	0	0	0	0	0	0
right	0	0	0	0	0	0

An example of an undirected graph that is built from the adjacency matrix is shown in Figure 4.4. We are using a depth-first algorithm to traverse the graph and search for all paths from the left flanking to the right flanking sequence:

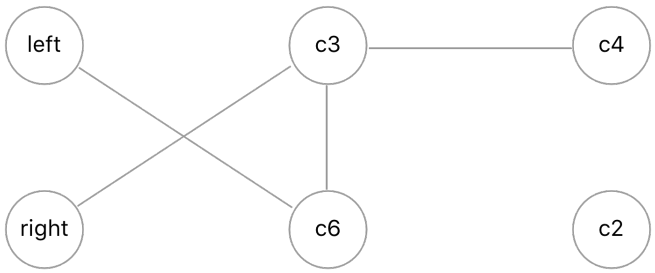

Figure 4.4: An undirected graph that represents overlaps between contigs and all possible paths from the left flanking to the right flanking for Alu chr10_9510994.

We merge all contigs for every found path and align the merged sequences to the Alu consensus database - a FASTA file that has Alu subtypes names and consensus sequences as shown in Figure 4.5. The BLAST tool is used again for this purpose. The merged sequence with the longest alignment to the consensus that exceeds the defined threshold is selected as the one that contains the full insertion sequence.

```
>ALU    SINE1/7SL       Primates
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcgggaggattgcttgagc
ccaggagttcgagaccagcctgggcaacatagcgagaccccgtctctacaaaaaatacaaaaattagccg
ggcgtggtggcgcgcgcctgtagtcccagctactcgggaggctgaggcaggaggatcgcttgagcccagg
agttcgaggctgcagtgagctatgatcgcgccactgcactccagcctgggcgacagagcgagaccctgtc
tcaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
>AluJo
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcgggaggattgcttgagcc
caggagttcgagaccagcctgggcaacatagcgagaccccgtctctacaaaaaatacaaaaattagccggg
cgtggtggcgcgcgcctgtagtcccagctactcgggaggctgaggcaggaggatcgcttgagcccaggagt
tcgaggctgcagtgagctatgatcgcgccactgcactccagcctgggcgacagagcgagaccctgtctca
>AluJb
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcgggaggatcacttgagcc
caggagttcgagaccagcctgggcaacatggtgaaaccccgtctctacaaaaaatacaaaaattagccggg
cgtggtggcgcgcgcctgtagtcccagctactcgggaggctgaggcaggaggatcgcttgagcccgggagg
tcgaggctgcagtgagccgtgatcgcgccactgcactccagcctgggcgacagagcgagaccctgtctca
```

Figure 4.5: **Example of the Alu consensus database.** The description line starts with the ">" sign and contains the subtype name. Then, the consensus sequence is reported.

In the final stage, the MEI information is extracted from the alignment data to the consensus database (see Figure 4.6). The subject name (identifier) is the subtype, i.e. AluYa5 in this case. The subject start and end alignment positions identify the length and the orientation of the insertion sequence - if the start is less than the end, then the sequence is reversed. We extract the actual MEI sequence from our full sequence using the start and end positions in the query. The rest of the sequence is considered to be flanking.

query id	subject id	% identity	length of alignment	# of mis-matches	# of gaps	query start	query end	subject start	subject end	expect value	bit score
left_flank_Contig6_Contig3_right_flank	AluYa5	99.29	283	1	1	637	919	282	1	2e-147	510

Figure 4.6: **Example of the alignment to the consensus.** The result of the alignment of the reconstructed insertion and the flanking sequences to the consensus database provides the information about the subtype of MEI (AluYa5), the percentage of identity (99.29%), the length of the alignment (283), the number of mismatches and gaps. As well, the start and the end positions in the query (our sequence) and subject (the consensus sequence) are reported. We can extract the actual sequence of the insertion by using these positions.

TSD is deduced from mapping the full sequence to the pre-integration reference sequence that contains both flanking regions using the previously described approach. An example of the alignment is shown in Figure 4.7.

```
# BLASTN 2.2.26 [Sep-21-2011]
# Query: left_flank_Contig5_Contig2_right_flank
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. start, s. end, e-value, bit score
left_flank_Contig5_Contig2_right_flank  chr10   99.83   601     1       0       1       601     1       601     0.0     1189
left_flank_Contig5_Contig2_right_flank  chr10   99.35   612     0       1       914     1521    589     1200    0.0     1187
```

Figure 4.7: **Example of the TSD alignment.** When aligning the full sequence to the reference genome, we can see the overlap at the end of the left flanking sequence and at the beginning of the right one (1 - 601 and 589 - 1,200). It indicates that there is a TSD that starts at 589 bp and ends at 601 bp in the extracted reference genome. Therefore, the actual sequence of TSD can be deduced from it.

The fully characterized MEI is reported as shown in Figure 4.8.

description of insertion allele

```
>1KP||ME|SINE:Alu:AluYa5|Strand|-|Genome|hg19|Pos|chr10:9510994-9510995|Allele|ins|Insertion|non
-ref|TSD|13:TGCAAATTTTTTT|IMD|0:|5TR|0:|3TR|35:TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
acttaaagtagctttccacattcacagagtctgttaagtggcagtcaagatttaaatctcagttgtctcatacttcagtctgttcttttaactacT
CTGCTAAAATTTGATGAGTTAGAATTGATTAGGATTCTGTACTCCTTATGGTAGAAAAATAAAATGAAAAAAGCTAAAATAGGAAAATTGATTTGT
ACCCTGATGGGATTTAATTTGACATGCTCTAAGCATTTGCATGATGGTAGACTTGACTGTTTATGTATATACATGTGTGAATGTGAATGGCTATAA
ATATAGAAGTAGAAGGCCTCTTTCTCCCTTTTGGTTTTGGTCATGTGGCTTTGCTCTAAAACCAGGTAAAACATTGGCAATATGCAGTATAGCTTA
GTATTAGCAACTAGAAATCACTTGGTAAAAATGAGTGGATCAATGACAATATTCACACACTAAATATCTGTTTACAATTCACATGCAGTGTGTACT
GACCACACATTGATTTTTCCTAATACATTTTGAACAAATATAAGTGAATTCCTATTTGAATATATACATATAAAATAAAATATGAAGGCCCTTGAT
GGCATACGAGGTGCAAATTTTTTT- left flanking sequence
TGCAAATTTTTTT - TSD                                                    insertion sequence
TGAGACGGAGTCTCGCTCTGTCGCCCAGGCTGGAGTGCAGTGGCGGGATCTCGGCTCACTGCAAGCTCTGCCTCCCGGGTTCACGCCATTCTCCTG
CCTCAGCCTCCCAAGTAGCTGGGACTACAGGCGCCCGCCACTACGCCCGGCTAATTTTTTTGTATTTTTAGTAGAGACGGGGTTTCACCGTTTTAG
CCGGGATGGTCTCGATCTCCTGACCTCGTGATCCGCCCGCCTCGGCCTCCCAAAGTGCTGGGATTACAGGCGTGAGCCACCGCGCCCGGCC
TGCAAATTTTTTT - TSD
TGCAAATTTTTTTAAGTGGAGTATTTTATGTTTGAATATAGAACTAAAGTAGAAGTTTCAGGGTTAGTTTTTTAAAAAAGTAAAATTTTAACTGAA
AGTCAGCCATTTTCTTTAAAAACATTTCAGTTTTTAAGAGATTCAATTGTTTTTGAATTAGCTGATGAAAAATGCCATGCAAATCGAAGTATCATT
AGCACTGCAATTTTTGTTTACTCTCATTATTCTGCTTTAACTCTCATATTATATAATCCTGAATATGTTACTTAAATATAGGTGACCTTGTGGAAA
ATTCAGTATCAGGTTTTATATCAAGGGTTCAATGTTGTGGGTATTGCTTTTAGTAGAGAAATCAACTCCTGTGGAAAAATTAAAAAAAAAAAAAG
GATGAGTATTCTCAGTCATTCTTTGCCTTTCTCAAACTTCTTTCCCTAAGGCAAAGTGTGAACCACACTTTCAGCTCTAAGAGCTATTACTAATTC
ATAGTACTTCCATTTACTCTTTTAACCAAGGATTGATAGTAATGCTGCTAATGAAATGTCCTGACCAGCTGTGGCTCAATTACTTGACATCGCTTC
TATTTACCTAGCCAGCTTCCGAAAGAACACCAA - right flanking sequence
```

description of pre-integration allele

```
>1KP||ME|SINE:Alu:AluYa5|Strand|-|Genome|hg19|Pos|chr10:9510994-9510995|Allele|pre|Insertion|non
-ref|TSD|13:TGCAAATTTTTTT|IMD|0:|5TR|0:|3TR|35:TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
acttaaagtagctttccacattcacagagtctgttaagtggcagtcaagatttaaatctcagttgtctcatacttcagtctgttcttttaactact
ctgctaAAATTTGATGAGTTAGAATTGATTAGGATTCTGTACTCCTTATGGTAGAAAAATAAAATGAAAAAAGCTAAAATAGGAAAATTGATTTGT
ACCCTGATGGGATTTAATTTGACATGCTCTAAGCATTTGCATGATGGTAGACTTGACTGTTTATGTATATACATGTGTGAATGTGAATGGCTATAA
ATATAGAAGTAGAAGGCCTCTTTCTCCCTTTTGGTTTTGGTCATGTGGCTTTGCTCTAAAACCAGGTAAAACATTGGCAATATGCAGTATAGCTTA
GTATTAGCAACTAGAAATCACTTGGTAAAAATGAGTGGATCAATGACAATATTCACACACTAAATATCTGTTTACAATTCACATGCAGTGTGTACT
GACCACACATTGATTTTTCCTAATACATTTTGAACAAATATAAGTGAATTCCTATTTGAATATATACATATAAAATAAAATATGAAGGCCCTTGAT
AGCATACGAGGTGCAAATTTTTTT- left flanking sequence
TGCAAATTTTTTT- TSD
AAGTGGAGTATTTTATGTTTGAATATAGAACTAAAGTAGAAGTTTCAGGGTTAGTTTTTTAAAAAAGTAAAATTTTAACTGAAAGTCAGCCATTTT
CTTTAAAAACATTTCAGTTTTTAAGAGATTCAATTGTTTTTGAATTAGCTGATGAAAAATGCCATGCAAATCGAAGTATCATTAGCACTGCAATTT
TTGTTTACTCTCATTATTCTGCTTTAACTCTCATATTATATAATCCTGAATATGTTACTTAAATATAGGTGACCTTGTGGAAAATTCAGTATCAGG
TTTTATATCAAGGGTTCAATGTTGTGGGTATTGCTTTTAGTAGAGAAATCAACTCCTGTGGaaaaatttaaaaaaaaaaaaaaaaaaGGATGAGTAT
TCTCAGTCATTCTTTGCCTTTCTCAAACTTCTTTCCCTAAGGCAAAGTGTGAACCACACTTTCAGCTCTAAGAGCTATTACTAATTCATAGTACTT
CCATTTACTCTTTTAACCAAGGATTGATAGTAATGCTGCTAATGAAATGTCCTGACCAGCTGTGGCTCAATTACTTGACATCGCTTCTATTTACCT
AGCCAGCTTCCGAAAGAACACCAA - right flanking sequence
//
```

Figure 4.8: The characterized Alu chr10_9510994 result.

4.2 Summary

In total, we processed 1,372 locations of Alu, 95 locations of SVA, and 146 locations of L1. Overall, we were able to characterize 62.76% of Alu events, 31.58% of SVAs, and 26.71% of L1s.

The LTR locations were not included in the characterization process due to long terminal repeats that occur on both sides of the insertion. These repeats can be around 1,000 bp in size. Thus our algorithm will consider them as overlapping contigs and merge them. We will address this problem in the future work by searching for specific sequence patterns in long terminal repeats that were described in the previous chapter. Currently, our tool can characterize only solo-LTR alleles.

To calculate the sensitivity, we run the tool on the list of non-reference MEIs from the Retrotransposon Insertion Polymorphism in Humans (dbRIP) data for which the full sequences are annotated. The dbRIP collected and compiled all available published data [75]. It accumulated the complete sequences of the non-reference alleles for a total of 3,194 loci covering Alu, SVA, L1, LTR, and HERVs, which represent only a small portion of non-reference MEIs existing in the human genomes. For each reported entry, they collected the following information: original identifications, type, association with disease, DNA sequences, target site duplications, 400 bp of flanking sequence regions, etc. This database was integrated into the UCSC genome browser created by the Genome Bioinformatics Group of University of California at Santa Cruz.

We compared the obtained insertion sequences from our output to the dbRIP database sequences and calculated the total success for each ME type. The results are represented in Table 4.2.

Table 4.2: dbRIP locations stats.

	Alu	SVA	L1
Overall locations processed	862	18	203
Successfully characterized	467	13	146
Total success rate	54%	72%	72%

One location for which the full insertion sequence is known was randomly selected to test different third-party tools and parameters as well as read quality thresholds and patterns that would result in better characterization of MEIs. The chosen location

corresponds to Alu entry and has high allele frequency. The sets of parameters and thresholds for third-party tools and read validation that were tested are represented in Appendix D. The main criterion was if the tool/threshold leads to a fully characterized sequence. We described the tools and parameters that worked best for our purpose in the previous chapter.

The numeric summary for 1kGP data for each MEI type is represented in Table 4.3.

On average for all three types, there are around 300,000 raw reads per location, and 1,497 of them are qualified. The average coverage is equal to 117x, which means each nucleotide is represented by 117 reads.

Table 4.3: MEI stats.

	Alu	SVA	L1
Overall locations processed	1,372	95	146
Successfully characterized	860	7	19
Partially characterized (with a gap in the middle)	2	23	20
Failed to be characterized	510	65	107
Sensitivity (total success rate)	62.76%	31.58%	26.71%
Raw reads per location	$\sim$ 300,000	$\sim$ 300,000	$\sim$ 300,000
Qualified reads per location	1040	1810	1642
Coverage	74	146	132

We observed that one of the reasons for failed characterization is low allele frequency. That means that the threshold of qualified reads that cover the regions of interest is not met. The second reason is the high repetitiveness of flanking sequences. In this case, we collect many false-positive reads that do not cover either the breakpoints sequences or the insertion part. Further investigation of these cases is necessary.

We introduced the frequency threshold for which our tool shows the highest success rate. We calculate the average amount of left split-reads, right split-reads, and discordant reads for each location for every MEI type separately. Then, we find the threshold for the mean of all reads by minimizing the number of false-negative values above the threshold while maximizing the number of true-positive ones above it. The

minimum required frequency and the corresponding success rate for each MEI type are represented in Table 4.4.

Table 4.4: Minimum required threshold – average amount of discordant, 5' and 3' end split-reads – and the corresponding success rate for each MEI type.

Type	Threshold	Success rate
Alu	94	75.03%
SVA	363	71.43%
L1	355	77.78%

The plots that represent the average amount of discordant and split-reads per location, the status of MEI (i.e. characterized or failed), and the optimal threshold are included in Appendix C.

The independent-samples t-test was conducted to test the hypothesis that there is a sufficient difference between data of the characterized and failed locations. The H_o hypothesis is that there is no difference between two sets of data.

The t-test results are represented in Table 4.5. Before applying t-test, the F-test was performed to determine equality of variance. In all cases, the variances are not equal, and we proceed with the t-test that assumes unequal variances.

Table 4.5: T-test results.

ME	Successful	Failed	p-value
Alu	199	113	0.002949542
SVA	383	159	0.001463012
L1	357	403	0.000375003

Since the values of the probability of random occurrence of the analyzed samples (e.g., $p = 0.002949542$ for Alu) is less than the significance level, which is $p = 0.05$, the null hypothesis is rejected. Consequently, differences between the samples are not random and they are considered significantly different from each other. Therefore, based on the t-test results we can conclude that the mean of discordant and split-reads influence the characterization.

The average time to run the system for one location consists of two parts: (i) the collection of qualified reads, which takes approximately 2.5 hours, and (ii) the

assembly and MEI information extraction, which takes up to one minute. However, there was one case where the number of valid contigs reached 279 and the time to align and assess the overlaps and process a graph took more than 10 hours.

To speed up the read collection phase, we can use the proposed threshold for discordant and split-reads. After reaching the determined amount of data, we can stop processing BAM files. This approach helped to reduce the required time by half only in some cases.

Moreover, the collection of reads part is the most memory consuming. At most 5 GB of heap size is required to run the system. Depending on the RAM of the machine where the tool is run, the allocated memory can be bigger just to provide some window.

Table 4.6: Computer resource usage per location.

Average collection of reads runtime	2.5 hours
Collection of reads memory usage	up to 5 GB
Average assembly runtime	1 minute
Average assembly memory usage	up to 2 GB
Intermediate output files	up to 5 MB

Chapter 5

Conclusions and Future Work

The sequence characterization of structural variants is still an open problem, especially for very long repetitive insertions. In this thesis, we proposed an approach for complete genome sequence characterization for insertional structural variants in human genomes. Our method is based on collecting discordant, concordant and split-reads from all publicly available human genome data that cover breakpoints and insertions. These reads are validated based on quality filters and patterns. Then, they are assembled into contigs using local *de novo* sequence assembly, and the contigs are merged from both sides of the insertion until they overlap. The information about a particular MEI is extracted from the alignment to the consensus, and TSD is deduced. In case of a successful characterization, the full insertion sequence is reported, otherwise, at least the breakpoint sequences are obtained. The developed system is the first tool that is aiming to provide the full sequence characterization of SVs.

Our tool is developed in the Java programming language and was run on SHARCNET clusters to parallelize and speed up the computations. The tool can accept different parameters that are listed in Table A.1. The requirements for running the system are described in Appendix A, and a full user manual with the link to the tool is provided in Appendix B. The tool is open-source and is available on GitHub by the link `https://github.com/YaroslavaGirilishena/me_builder`.

Experimental results with dbRIP data showed that we were able to achieve a sensitivity value of 54% for Alu and 72% for SVA and L1. The success rate varies among ME types in 1KGP data as well. The best sensitivity value of 75.03% was achieved for Alu characterization, for SVA it is 71.43%, and for L1 it is 77.78% when the frequency threshold was introduced.

Evaluating the correctness of the generated sequence for non-reference MEIs is a

difficult problem since there is no reference genome against which to compare. Due to the diversity of human genomes, the overlaps between created contigs are difficult to assess. Therefore, we allow a higher percentage of mismatches than commonly accepted for reads assembly.

The major problems that we faced were the low allele frequency and the high repetitiveness of the flanking regions. The strategies to overcome these potential pitfalls need further investigation.

The major improvements that can be made in order to characterize the longest insertions are as follows:

1. Research patterns for longer insertions when aligning contigs against each other and when merging contigs into scaffolds.

2. The alignment of the scaffolds to the consensus database has to be improved by either developing a local alignment algorithm that will consider poly A/T diversity and mutations or researching and adjusting other existing tools. The validation approach of the alignment to the consensus database for different types of MEIs has to be improved as well to consider the distinctions in the sequence structure.

3. For very long MEIs, collect concordant reads that cover the insertion part by analyzing the depth of coverage of reads in the specific regions of the genome. Then, do the "bridge" assembly using the long already constructed scaffolds and concordant read pairs.

4. Improve the assembly part by aligning contigs that are not merged with the flanking regions to the MEI consensus sequences. Potentially, they can be aligned to the middle of the insertion, thus covering the gap. Then using concordant reads that are mapped to the insertion we can extend the contig in the middle to the flanking regions while extending the flankings until they all overlap.

5. Construct a database of intermediate output for better assessment of the results. A large number of failed results contain the full insertion merged with one flanking region. Usually, there are only 10-20 bases left to join the insertion to the other flanking. More detailed analysis of the intermediate output can reveal the reason why in this particular case the two sequences are not connected. Moreover, such small gaps in the insertions could be reconstructed manually.

The main objective of the future work is to integrate these tools into a pipeline to streamline the personal genome analysis using the database of annotated and characterized SVs. The advantages of detailed sequence characterization in further genome analysis include, but are not limited to:

1. The detection and characterization of known SVs will be a straightforward task by mapping discordant reads as concordant reads to the structural variant sequences.

2. The time-consuming validation of SVs can be eliminated if the sequence characterization is complete and accurate.

3. The analysis of novel SVs can be more efficient, when discordant reads that are associated with common SVs will be removed from the process.

As a longer term impact, the full characterization of structural variants can improve the accuracy of personal genome analysis, which in turn will benefit precision medicine and accelerate future genome studies.

Bibliography

[1] 1000 Genomes Project Consortium (2010). *The 1000 Genomes Project: SUPPLEMENTARY INFORMATION*.

[2] 1000 Genomes Project Consortium (2015). A global reference for human genetic variation. *Nature*, 526.

[3] Abyzov, A. and Gerstein, M. (2011). AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision. *Bioinformatics*, 27(5):595–603.

[4] Abyzov, A., Urban, A., Snyder, M., and Gerstein, M. (2011). CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Research*, 21(6):974–984.

[5] Adams, M. et al. (2000). The genome sequence of Drosophila melanogaster. *Science*, 287:2185–2195.

[6] Alkan, C., Coe, B., and Eichler, E. (2011). Genome structural variation discovery and genotyping. *Nature Reviews Genetics*, 12:363–376.

[7] Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410.

[8] Anderson, S. (1981). Shotgun DNA sequencing using cloned DNase I-generated fragments. *Nucleic Acids Research*, 9(13):3015–3027.

[9] Ashlock, W. (2015). CIBCB 2015 Tutorial Bioinformatics of epigenome. York University, Toronto, Canada. `https://creativecommons.org/licenses/by-sa/4.0/`, accessed on August, 2017.

[10] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A., Dvorkin, M., Kulikov, A., Nikolenko, S., Pham, S., Prjibelski, A., Pyshkin, A., Sirotkin, A., Vyahhi, N.,

Tesler, G., Alekseyev, M., and Pevzner, P. (2012). SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Combinatorial Biology*, 19:455–477.

[11] Bentley, D. (2006). Whole-genome re-sequencing. *Current Opinion in Genetics & Development*, 16:545–552.

[12] Campbell, P. et al. (2008). Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nature Genetics*, 40(6):722–729.

[13] Chaisson, M., Brinza, D., and Pevzner, P. (2009). De novo fragment assembly with short mate-paired reads: does the read length matter? *Genome Research*, 19:336–346.

[14] Chaisson, M. and Pevzner, P. (2008). Short read fragment assembly of bacterial genomes. *Genome Research*, 18:324–330.

[15] Chen, K. et al. (2009). BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nature Methods*, 6:677–681.

[16] Chiang, D., Getz, G., Jaffe, D., O'Kelly, M., Zhao, X., Carter, S., Russ, C., Nusbaum, C., Meyerson, M., and Lander, E. (2009). High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nature Methods*, 6(1):99–103.

[17] Clair, C. S. and Visick, J. (2010). *Exploring Bioinformatics: A Project-Based Approach 2nd*. Jones and Bartlett Publishers, Inc.

[18] Dohm, J., Lottaz, C., Borodina, T., and Himmelbauer, H. (2007). SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17(11):1697–1706.

[19] EMBL-EBI (2008). Velvet: Sequence assembler for very short reads. `https://www.ebi.ac.uk/~zerbino/velvet/`, accessed on August, 2017.

[20] Hajirasouliha, I., Hormozdiari, F., Alkan, C., Kidd, J., Birol, I., Eichler, E., and Sahinalp, S. (2010). Detection and characterization of novel sequence insertions using paired-end next-generation sequencing. *Bioinformatics*.

[21] Haraksingh, R. and Snyder, M. (2013). Impacts of variation in the human genome on gene regulation. *Journal of Molecular Biology*, 425:3970–3977.

[22] Hormozdiari, F., Alkan, C., Eichler, E., and Sahinalp, S. (2009). Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Research*, 19(7):1270–1278.

[23] Hormozdiari, F. et al. (2010). Next-generation VariationHunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*.

[24] Hormozdiari, F., Hajirasouliha, I., McPherson, A., Eichler, E., and Sahinalp, S. (2011). Simultaneous structural variation discovery in multiple paired-end sequenced genomes. *Genome Research*, 21(12):2203–2212.

[25] Huang, X. and Madan, A. (1999). CAP3: A DNA sequence assembly program. *Genome Research*, 9:868–877.

[26] Iafrate, A., Feuk, L., Rivera, M., Listewnik, M., Donahoe, P., Qi, Y., Scherer, S., and Lee, C. (2004). Detection of large-scale variation in the human genome. *Nature Genetics*, 36:949–951.

[27] Idury, R. and Waterman, M. (1995). A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2:291–306.

[28] Institute, B. Decoding SAM flags. `http://broadinstitute.github.io/picard/explain-flags.html`, accessed on August, 2017.

[29] Institute, B. G. (2010). Short oligonucleotide analysis package. `http://soap.genomics.org.cn/soapdenovo.html`, accessed on August, 2017.

[30] Jeck, W., Reinhardt, J., Baltrus, D., Hickenbotham, M., Magrini, V., Mardis, E., Dangl, J., and Jones, C. (2007). Extending assembly of short DNA sequences to handle error. *Bioinformatics*.

[31] Kazazian, H. J. et al. (2004). Mobile elements: Drivers of genome evolution. *Science*, 303(1626).

[32] Kazazian, H. J., Wong, C., Youssoufian, H., Scott, A., Phillips, D., and Antonarakis, S. (1988). Haemophilia A resulting from de novo insertion of L1 sequences represents a novel mechanism for mutation in man. *Nature*, 332(6160):164–166.

[33] Keane, T., Wong, K., and Adams, D. (2013). RetroSeq: transposable element discovery from next-generation sequencing data. *Bioinformatics*, 29(3):389–390.

[34] Kehr, B., Melsted, P., and Halldorsson, B. (2016). Popins: population-scale detection of novel sequence insertions. *Bioinformatics*, 32(7):961–967.

[35] Kent, J. BLAT search genome. `http://genome.ucsc.edu`, accessed on August, 2017.

[36] Korbel, J., Urban, A., Affourtit, J., Godwin, B., Grubert, F., Simons, J., Kim, P., Palejev, D., Carriero, N., Du, L., Taillon, B., Chen, Z., Tanzer, A., Saunders, A., Chi, J., Yang, F., Carter, N., Hurles, M., Weissman, S., Harkins, T., Gerstein, M., Egholm, M., and Snyder, M. (2007). Paired-end mapping reveals extensive structural variation in the human genome. *Science*, 318(5849):420–426.

[37] Kubalik, J., Buryan, P., and Wagner, L. (2010). Solving the DNA fragment assembly problem efficiently using iterative optimization with evolved hypermutations. *GECCO '10 Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 213–214.

[38] Lander, E. et al. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409:860–921.

[39] Lee, E. et al. (2013). Landscape of somatic retrotransposition in human cancers. *Science*, 337:967–971.

[40] Li, D., Liu, C., Luo, R., Sadakane, K., and Lam, T. (2015). MEGAHIT: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, 31(10):1674–1676.

[41] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., and Subgroup, . G. P. D. P. *Sequence Alignment/Map format Specification*. `https://github.com/samtools/hts-specs`, accessed on August, 2017.

[42] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., and Subgroup, . G. P. D. P. (2009a). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079.

[43] Li, R. et al. (2009b). Building the sequence map of the human pan-genome. *Nature Biotechnology*, 28:57–63.

[44] Li, R. et al. (2009c). De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20:265–272.

[45] Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., Tang, J., Wu, G., Zhang, H., Shi, Y., Liu, Y., Yu, C., Wang, B., Lu, Y., Han, C., Cheung, D., Yiu, S., Peng, S., Xiaoqian, Z., Liu, G., Liao, X., Li, Y., Yang, H., Wang, J., Lam, T., and Wang, J. (2012). SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*.

[46] Manual, U. U. O. U. (2014). CAP3. `https://ugene.net/wiki/display/UUOUM15/CAP3`, accessed on September, 2017.

[47] Mardis, E. (2008). Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics*, 9:387–402.

[48] Mardis, E. and Wilson, R. (2009). Cancer genome sequencing: a review. *Human Molecular Genetics*.

[49] Margulies, M. et al. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380.

[50] Marx, V. (2015). The DNA of a nation. *Nature*, 524:503–505.

[51] Maxam, A. and Gilbert, W. (1977). A new method for sequencing DNA. *Proceedings of the National Academy of Sciences of the United States of America*, 74:560–564.

[52] McCarroll, S. et al. (2008a). Integrated detection and population-genetic analysis of SNPs and copy number variation. *Nature Genetics*, 40(10):1166–1174.

[53] McCarroll, S., Huett, A., Kuballa, P., Chilewski, S., Landry, A., Goyette, P., Zody, M., Hall, J., Brant, S., Cho, J., Duerr, R., Silverberg, M., Taylor, K., Rioux, J., Altshuler, D., Daly, M., and Xavier, R. (2008b). Deletion polymorphism upstream of IRGM associated with altered IRGM expression and Crohn's disease. *Nature Genetics*, 40(9):1107–1112.

[54] McCarthy, E. and McDonald, J. (2003). LTR STRUC: a novel search and identification program for LTR retrotransposons. *Bioinformatics*, 19(3):362–367.

[55] Medvedev, P., Stanciu, M., and Brudno, M. (2009). Computational methods for discovering structural variation with next-generation sequencing. *Nature Methods*.

[56] Miki, Y., Katagiri, T., Kasumi, F., Yoshimoto, T., and Nakamura, Y. (1996). Mutation analysis in the BRCA2 gene in primary breast cancers. *Nature Genetics*, 13(2):245–247.

[57] Mouse Genome Sequencing Consortium (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562.

[58] National Center for Biotechnology Information. Standard nucleotide BLAST. `https://blast.ncbi.nlm.nih.gov/Blast.cgi`, accessed on August, 2017.

[59] Onishi-Seebacher, M. and Korbel, J. (2011). Challenges in studying genomic structural variant formation mechanisms: The short-read dilemma and beyond. *Bioessays*, 33(11):840–850.

[60] Ono, M., Kawakami, M., and Takezawa, T. (1987). A novel human nonviral retroposon derived from an endogenous retrovirus. *Nucleic Acids Res*, 15:8725–8737.

[61] Orengo, C., Jones, D., and Thornton, J. (2003). *Bioinformatics: gene, proteins and computers*. BIOS Scientific Publishers Limited.

[62] Ostertag, E., Goodier, J., Zhang, Y., and Kazazian, H. J. (2003). SVA elements are nonautonomous retrotransposons that cause disease. *The American Journal of Human Genetics*, 73(6):1444–1451.

[63] Pevzner, P. (2000). *Computational Molecular Biology: An Algorithmic Approach*. The MIT Press, Cambridge, Massachusetts, London, England.

[64] Pevzner, P., Tang, H., and Tesler, G. (2004). De novo repeat classification and fragment assembly. *Genome Research*, 14:1786–1796.

[65] Pevzner, P., Tang, H., and Waterman, M. (2001). An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 98:9748–9753.

[66] Qian, Y., Kehr, B., and Halldorsson, B. (2015). PopAlu: population-scale detection of Alu polymorphisms. *PeerJ*.

[67] Rausch, T., Zichner, T., Schlattl, A., Stutz, A., Benes, V., and Korbel, J. (2012). DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28.

[68] Simpson, J., Wong, K., Jackman, S., Schein, J., Jones, S., and Birol, I. (2009). ABySS: a parallel assembler for short read sequence data. *Genome Research*, 19:1117–1123.

[69] Snyder, M. (2010). Structural variation in the human genome. Stanford University. `http://biochem158.stanford.edu`.

[70] Stewart, C. et al. (2011). A comprehensive map of mobile element insertion polymorphisms in humans. *PLOS Genetics*.

[71] Tatusova, T. and Madden, T. (1999). BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters*.

[72] Thung, D., de Ligt, J., Vissers, L., Steehouwer, M., Kroon, M., de Vries, P., Slagboom, E., Ye, K., Veltman, J., and Hehir-Kwa, J. (2014). Mobster: accurate detection of mobile element insertions in next generation sequencing data. *Genome Biology*, 15.

[73] Tuzun, E. et al. (2005). Fine-scale structural variation of the human genome. *Nature Genetics*, 37:727–732.

[74] Venter, J. et al. (2001). The sequence of the human genome. *Science*, 291(5507):1304–1351.

[75] Wang, J., Song, L., Grover, D., Azrak, S., Batzer, M., and Liang, P. (2006). dbRIP: A highly integrated database of Retrotransposon Insertion Polymorphisms in humans. *Human Mutation*, 27:323–329.

[76] Warren, R., Sutton, G., Jones, S., and Holt, R. (2007). Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501.

[77] Wu, J., Lee, W., Ward, A., Walker, J., Konkel, M., Batzer, M., and Marth, G. (2014). Tangram: a comprehensive toolbox for mobile element insertion detection. *BMC Genomics*.

[78] Xing, J., Zhang, Y., Han, K., Salem, A., Sen, S., Huff, C., Zhou, Q., Kirkness, E., Levy, S., Batzer, M., and Jorde, L. (2009). Mobile elements create structural variation: analysis of a complete human genome. *Genome Research*, 19:1516–1526.

[79] Ye, K., Schulz, M., Long, Q., Apweiler, R., and Ning, Z. (2009). Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25:2865–2871.

[80] Yngvadottir, B., Macarthur, D., Jin, H., and Tyler-Smith, C. (2009). The promise and reality of personal genomics. *Genome Biology*, 10(237).

[81] Zerbino, D. (2009). *Genome assembly and comparison using de Bruijn graphs*. Phd dissertation, Darwin College.

[82] Zerbino, D. and Birney, E. (2008). Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*.

[83] Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. (2000). A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology*, 7:203–214.

Appendix A

Input parameters and System requirements

The Java Virtual Machine requires at least 5 GB of heap size to run the program. Therefore, the parameter *-Xmx5g* should be included in the command line. The command to execute the program is as follows: *java -Xmx5g -d64 -jar MEBuilder.jar*

To run the tool on SHARCNET, the memory per process value should be at least twice the size of the allocated minimum heap size.

Table A.1: Required CLI parameters.

Parameter	**Description**	**Value**
-ME	type of MEI	Alu / SVA / L1
-i	input file with locations	.bed or .vcf format
-BAMpath	path for BAM files, required	/wgs/human/BAM
-BAMfile	BAM file name, if not specified, all BAM files in BAMpath will be used	/data/bam/ACB70G_p10.bam (the indexed ACB70G_p10.bai file should also be present in the same directory)
-SAMTOOLSpath	path for SAMtools executable, required	/usr/local/bin
-BLASTpath	path for blastn executable, required	/usr/local/bin
-BLASTdb	path and name of blast DB, required	/data/blastDB/MEIs/Alu.fa
-BL2SEQpath	path for bl2seq executable, required	/usr/local/bin
-CDHITPath	path to CD-HIT-EST executable, required	/usr/local/bin
-CAP3path	path for CAP3 executable, required	/usr/local/bin

Table A.2: Optional CLI parameters.

Parameter	**Description**	**Value**
-c	chromosome name	chr1-chr22, chrX, chrY
-p	position in the chromosome	$[0, 2^{31} - 1]$
-min_ins_length	minimum length of insertion alignment to the consensus DB	50-200
-dev	option to use default parameters from the development mode on SHARCNET	1
-config	configuration file with all input parameters described above	./config.properties

Appendix B

User Manual

B.1 Download and Installation

The tool is open-source and is available on GitHub by the link `https://github.com/YaroslavaGirilishena/me_builder`. Contribution is welcomed. To download it, follow the GitHub guidance.

B.2 Third-party tools

There is a list of required third-party tools that have to be installed before running the system. The paths to the executable files to each third-party tool have to be provided as input as described further.

1. **SAMtools** is essential for processing .bam files and collecting reads. To download and install the tool, follow the link: `https://github.com/samtools/samtools`.

2. **CD-HIT** is used to remove redundant reads before the assembly by running the CD-HIT-EST program. Follow the link to download the tool: `https://github.com/weizhongli/cdhit`.

3. **CAP3** tool is used to assemble the reads into contigs. The link to download: `http://seq.cs.iastate.edu/cap3.html`.

4. **BLAST** program *blastn* is integrated to perform the alignment of the contigs to the consensus database. To download and install the tool, follow the link: `https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download` and `https://www.ncbi.nlm.nih.gov/books/NBK279671/`.

5. **bl2seq** program does the all-against-all pairwise alignment of contigs which is necessary for the "bridge" assembly of the insertion sequence. The program is distributed with the blastall program by the NCBI. The link to download is: `https://blast.ncbi.nlm.nih.gov/Blast.cgi`.

B.3 How to run

The minimum required heap size that has to be allocated to run the system is 5 GB. This value depends on the size of a .bam file that contains raw reads. 5 GB was enough to process a 70 GB .bam file.

The command to execute the program is:

java -Xmx5g -d64 -jar MEBuilder.jar

Where -Xmx5g is a parameter for memory allocation, and MEBuilder.jar is the executable file.

For more information about Java command line execution, please refer to `http://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html`

B.4 Input

The list of the required and optional input parameters is:

- **-ME** - type of MEI. One of Alu / SVA / L1.
- **-c** - chromosome name - chr1-chr22, chrX, chrY. The parameter is used for running the system for one particular loci. It is an optional parameter, but is required when an input file with locations is not provided.
- **-p** - position in the chromosome, can take values from the range $[0, 2^{31} - 1]$. It is an optional parameter, but is required when an input file with locations is not provided.
- **-i** - input file with MEI locations. The acceptable file formats are .bed or .vcf. The parameter is required, but can be substituted by specifying chromosome and position parameters.
- **-min_ins_length** - the minimum length of insertion alignment to the consensus database. The input values must be in the range [50, 200].

- **-BAMpath** - full path to the folder with BAM files. Required. Can be substituted by -BAMfile when only one .bam is needed. For example, /wgs/human/BAM.
- **-BAMfile** - full BAM file path. Optional, if -BAMpath is specified. For example, /data/bam/ACB70G_p10.bam. The .bam files have to be indexed ahead, and the indexed ACB70G_p10.bai file should also be present in the same directory.
- **-SAMTOOLSpath** - path to SAMtools executable. Required. For example, /usr/local/bin.
- **-BLASTpath** - path for blastn executable, required. For example, /usr/local/bin.
- **-BLASTdb** - full path to the blast database, required. The consensus database has to be in a FASTA format following the template: each sequence has a description (i.e. name of subtype) line above that starts with the ">" symbol. Then, the actual sequence is provided. There are no extra spaces or symbols between sequences. Before working with the consensus database, it will be indexed automatically.
- **-BL2SEQpath** - path to bl2seq executable, required. For example, /usr/local/bin.
- **-CDHITPath** - path to CD-HIT-EST executable, required. For example, /usr/local/bin.
- **-CAP3path** - path to CAP3 executable, required. For example, /usr/local/bin.
- **-startLoci** - start location in the list of all events' locations, optional.
- **-endLoci** - end location in the list of all events' locations, optional.
- **-config** - configuration file with all input parameters described above. This parameter is optional, but if used, has to be the first and the only one. The file has to be in a .properties format. The template is included in the package.

B.5 Extra requirements

The system creates intermediate output that is used by later steps. For that reason, a minimum of 2 GB extra space has to be provided.

The intermediate output contains such directories:

1. **disc_reads/** folder that contains qualified reads in a FASTA format and the BAM folder with raw data of qualified reads in .txt format for studying purposes. To speed up the process when running the tool a second time do not delete this folder.

2. **intermediate_output/cap3_assembly** folder contains the output from running CAP3 tool.

3. **intermediate_output/contigs_for_merging** folder contains valid contigs parsed into separate files.

4. **intermediate_output/bl2seq_output** folder stores pairwise alignments data for all contigs pairs for each location.

5. **intermediate_output/bl2seq_output_flanking** folder contains the alignment data of contigs to the reference genome (specifically to the region of interest - 600 bp before and after the event point).

6. **intermediate_output/merged_contigs** folder stores the intermediate merged sequences of contigs and flanking regions.

7. **intermediate_output/ref_flanking** directory stores the reference genome sequence of the region of interest (600 bp before and after the insertion point).

8. **intermediate_output/tsd_alignment** folder contains the alignment data for TSD calculation.

9. **log/** where all logs are saved.

10. **results/** where all results are stored based on type, location and status: characterized, partially characterized (has a gap in the middle which is represented by Ns), and failed. It is best to remove or clean this folder before running the tool again.

Inside each of the described folders, except for disc_reads, each type has its own directory and each location has its own folder as well.

B.6 Package

In the additional package, some input data is included such as:

- The consensus database for covered types - /input/consensus.
- The .bed files with locations for each type - /input/data1KP/ and /input/non_reference.
- The collected discordant and split-reads for input locations - /disc_reads.
- The version of the reference genome that was used - /input/ref/hg19.

Appendix C

Plots

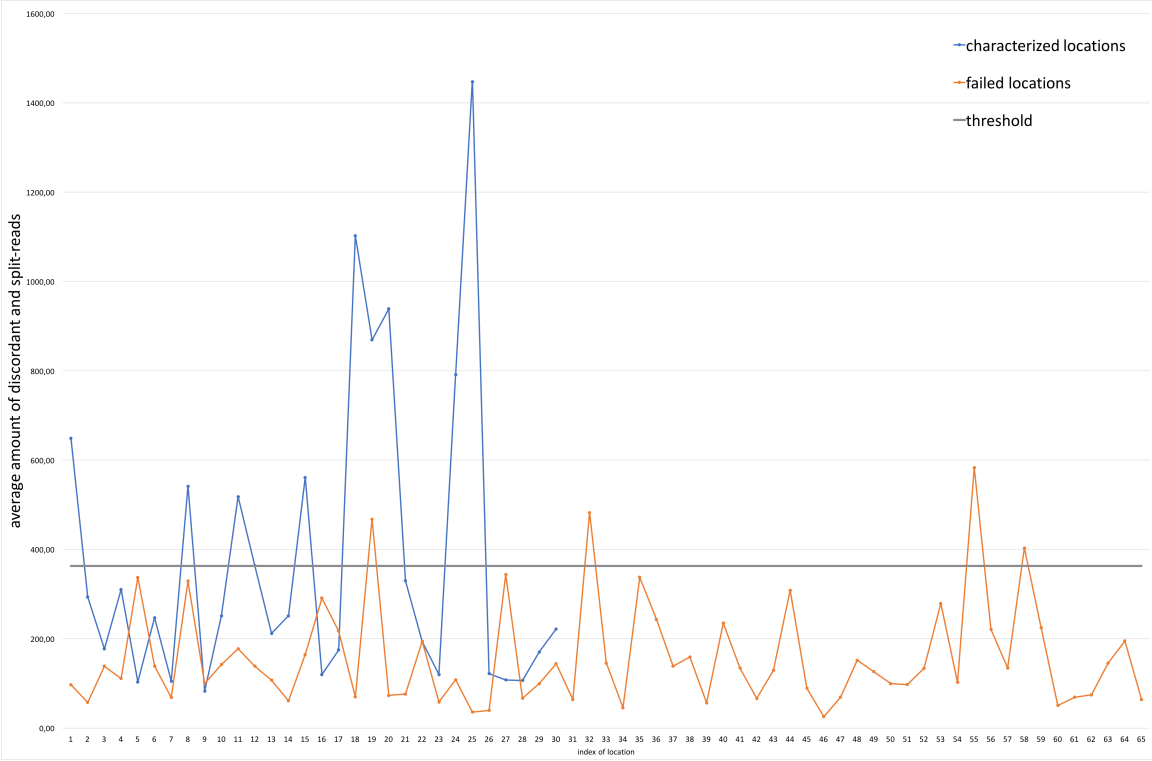

Figure C.1: **Average amount of discordant and split-reads for characterized and failed SVA locations with introduced threshold.** The blue dots that are connected by the blue line represent the mean of discordant and split-reads for MEIs that were characterized. The orange dots that are linked by the orange line represent the mean of discordant and split-reads for failed MEIs. The gray line is the optimal threshold for the average amount of discordant and split-reads that guarantees the highest success rate for SVA for our program.

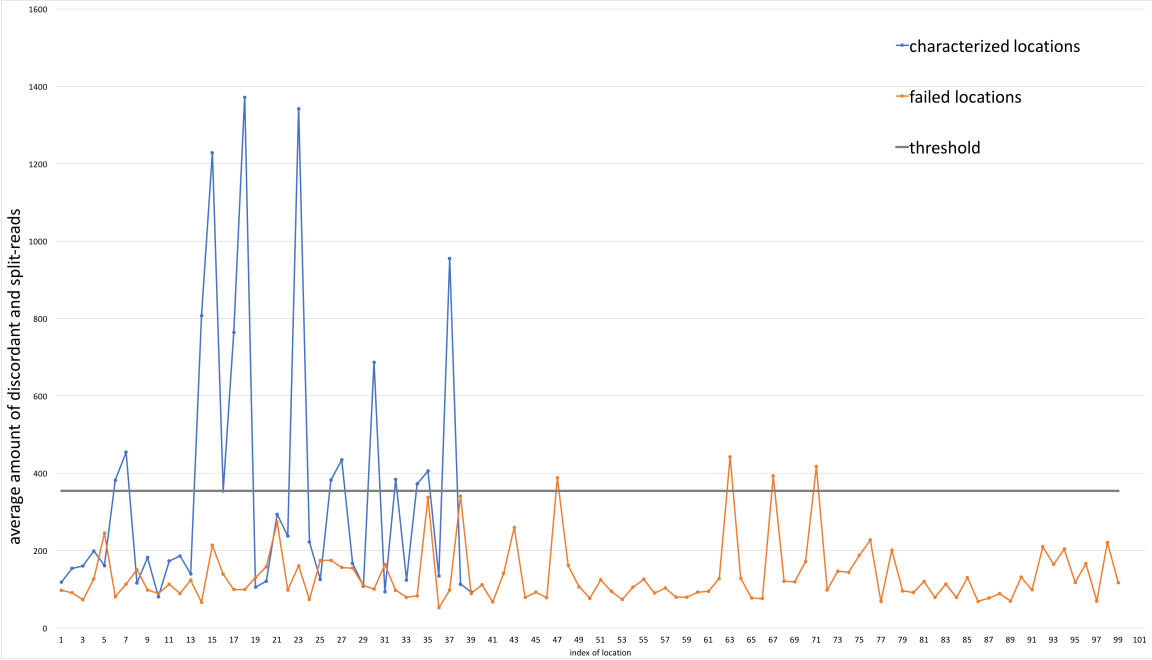

Figure C.2: **Average amount of discordant and split-reads for characterized and failed L1 locations with introduced threshold.** The blue dots that are connected by the blue line represent the mean of discordant and split-reads for MEIs that were characterized. The orange dots that are linked by the orange line represent the mean of discordant and split-reads for failed MEIs. The gray line is the optimal threshold for the average amount of discordant and split-reads that guarantees the highest success rate for L1 type for our program.

Figure C.3: **Average amount of discordant and split-reads for characterized and failed Alu locations with introduced threshold.** The blue dots that are connected by the blue line represent the mean of discordant and split-reads for MEIs that were characterized. The red dots that are linked by the red line represent the mean of discordant and split-reads for failed MEIs. The gray line with orange boxes is the optimal threshold for the average amount of discordant and split-reads that guarantees the highest success rate for Alu type for our program.

Appendix D

Experiments

D.1 Assembly tools

- CAP3

Table D.1: The CD-HIT and CAP3 parameters.

Tool	Parameter	Description
CD-HIT-EST	-c 0.98	percentage of identity
CAP3	-o 16	overlap length cutoff (in base pairs)
CAP3	-p 90	overlap percent identity cutoff
CAP3	-z 1	number of good reads at clip position (small because we removed redundant reads)

We found it convenient to specify the overlap percentage threshold and overlap length to overcome the sequence diversity and sequencing errors. Therefore, the CAP3 tool was selected as a primarily assembly tool. The description of the usage and more parameters options can be found in [46].

- Velvet [82]

The Velvet tool consists of two programs that have to be executed sequentially: (i) velveth takes files with reads and produces a hashtable, and (ii) velvetg that works with the output files from velveth and where the de Bruijn graph is built then manipulated.

Table D.2: Velvet parameters.

Program	Parameter	Description
velveth	21	hash length k
velveth	-fasta	input file format
velveth	-shortPaired	read type
velvetg	-min_contig_lgth 200	minimum length of contigs to keep
velvetg	-read_trkg yes	using read tracking
velvetg	-scaffolding no	scaffold contigs that cannot be connected

For a detailed description of parameters and different parameters options, refer to [19].

- SOAPdenovo [45]

  The SOAPdenovo assembler takes as input a configuration file with main parameters specified in there.

Table D.3: SOAPdenovo config file.

Parameter	Value	Description
avg_ins	200	the average insert size of the library
reverse_seq	0	if the read sequences need to be complementarily reversed
asm_flags	3	in which part(s) the reads are used. In this case in both contig and scaffold assembly
rd_len_cutof	100	assembler will cut the reads from the current library to this length
rank	1	in which order the reads are used for scaffold assembly
pair_num_cutoff	3	the cutoff value of pair number for a reliable connection between two contigs or pre-scaffolds
map_len	32	the minimum alignment length between a read and a contig required for a reliable read location

When the config file is ready, we run the *all* program that includes all steps involved in the assembly process.

Table D.4: SOAPdenovo parameters.

Program	Parameter	Description
all	-s config_file	a config file of reads
all	-K 13	k-mer size (min 13, max 63/127)
all	-R	resolve repeats by reads

For more parameters options, refer to [29].

The assembled contigs produced by Velvet and SOAPdenovo tools were comparatively short and had insufficient overlaps in the middle part of an insertion sequence. Moreover, the poly T/A tail with a variable number of nucleotides caused problems

during the assembly. It is not excluded that these assemblers can be used for contigs assembly part in our system. However, more research on input parameters is required which was not our primary focus in this thesis. The experiments with different assemblers are left for future work.

D.2 Sequencing quality

The initial set of sequencing quality thresholds is represented in Table D.5.

Table D.5: Set of default sequencing quality thresholds.

Parameter	Value	Description
MIN_BASE_QUAL	26	minimal quality for any individual base
MIN_AVG_READ_QUAL	28	minimal average quality values across the entire read
PERCENT_BASE_ ABOVE_QUAL	90	percentage of bases meeting above MIN_BASE_QUAL
MIN_NUM_OF_ BASES_ABOVE_QUAL	48	minimal number of bases with quality above MIN_BASE_QUAL
MIN_READ_LENGTH	50	minimal read length

We adjusted thresholds to collect more reads for the assembly part since the initially assembled contigs did not cover well the breakpoint and insertion sequences. We included a boolean parameter that indicates whether or not to use sequencing quality thresholds.

Table D.6: Set of adjusted sequencing quality thresholds.

Parameter	Value	Description
MIN_BASE_QUAL	10	minimal quality for any individual base
MIN_AVG_READ_QUAL	20	minimal average quality values across the entire read
PERCENT_BASE_ ABOVE_QUAL	95	percentage of bases meeting above MIN_BASE_QUAL
MIN_NUM_OF_ BASES_ABOVE_QUAL	48	minimal number of bases with quality above MIN_BASE_QUAL
MIN_READ_LENGTH	50	minimal read length